**News for Speed Demons
The 8Mhz Zip Chip is shipping
Send for yours today!
$157 plus $2 postage**

**The COMPUTIST BBS is On-line
at (206) 581-9292**

**COMPUTIST is Moving to:
33821 E Orville Rd.
Eatonville, WA 98328**

Readers Data EXchange

New COMPUTIST readers using Apple IIs are advised to read this page carefully to avoid frustration when attempting to follow a softkey or entering the programs printed in this issue.

What is a softkey, anyway?

Softkey is a term which we coined to describe a procedure that removes, or at least circumvents, any copy-protection on a particular disk. Once a softkey procedure has been performed, the resulting backup copy can usually be copied by the normal copy programs (for example: COPYA, on the DOS 3.3 System Master disk).

Commands and control keys

Commands which a reader is required to perform are set apart by being in boldface and on a separate line. The return key must be pressed at the end of every such command unless otherwise specified. Control characters are preceeded by "ctrl". An example of both is:
6 ctrl P

Type **6**. Next, place one finger on the ctrl key and then press **P**. Don't forget to press the return key.

Other special combination keypresses include ctrl reset and open-apple ctrl reset. In the former, press and hold down the ctrl key then press the reset key. In the latter, press and hold down both ctrl and open-apple then press reset .

Software recommendations

The Starter Kit contains most of the programs that you need to "Get started". In addition, we recommend that you acquire the following:

• Applesoft program editor such as "Global Program Line Editor (GPLE)".
• Assembler such as "Merlin/Big Mac".
• Bit-copy program such as "Copy II Plus", "Locksmith" or "Essential Data Duplicator".
• Word-processor (such as AppleWorks).
• "COPYA", "FID" and "MUFFIN" from the DOS 3.3 System Master disk.

Super IOB and Controllers

This powerful deprotection utility (in the COMPUTIST Starter Kit) and its various Controllers are used in many softkeys. (It is also on each Super IOB Collection disk.)

Reset into the Monitor

Softkeys occasionally require the user to stop the execution of a copy-protected program and directly enter the Apple's system monitor. Check the following list to see what hardware you will need to obtain this ability.

Laser 128: Your ROM includes a forced jump to the monitor. Press **ctrl return reset**.

Apple II+, //e, compatibles: 1) Place an Integer BASIC ROM card in one of the Apple slots. 2) Use a non-maskable interrupt (NMI) card such as Replay or Wildcard.

Apple II+, compatibles: 1) Install an F8 ROM with a modified reset-vector on the computer's motherboard as detailed in the "Modified ROM's" article (COMPUTIST #6 or Book Of Softkeys III) or the "Dual ROM's" article (COMPUTIST #19).

Apple //e, //c: Install a modified CD ROM on the computer's motherboard that changes the open-apple ctrl reset vector to point to the monitor. *(This will void an Apple //c warranty since you must open the case to install it.)*

Apple //gs: If you have the 2.x ROM, there is a hidden classic desk accessory (CDA) that allows you to enter the monitor. In order to install the new CDA, you should enter the monitor (CALL -151) before running any protected programs and press **# return** . This will turn on two hidden CDAs, Memory Peeker and Visit Monitor. Thereafter press **open-apple ctrl esc** to go to the Desk Accessories menu. Select Visit Monitor and there you are. Use **ctrl Y** to exit.

Recommended literature

• Apple II Reference Manual (or IIe, IIc, etc.)
• DOS 3.3 or ProDOS manual
• Beneath Apple DOS & Beneath Apple ProDOS, by Don Worth and Pieter Lechner, from Quality Software

Typing Applesoft programs

BASIC programs are printed in a format that is designed to minimize errors for readers who key in these programs. If you type:
`10HOME:REMCLEAR SCREEN`

The LIST will look like:

`10  HOME : REM CLEAR SCREEN`

Applesoft inserts spaces into a program listing before and after every command word or mathematical operator. These spaces don't pose a problem except when they are inside of quotes or after a DATA command. There are two types of spaces: those that have to be keyed and those that don't. Spaces that must be typed appear in COMPUTIST as special characters (◊). All other spaces are there for easier reading.

NOTE: If you want your checksums to match, only type spaces within quotes or after DATA statements if they are shown as (◊) charactors. SAVE the program at periodic intervals using the name given in the article. All characters after a REM are not checked by the checksum program so typing them is optional.

Typing Hexdumps

Machine language programs are printed in COMPUTIST as hexdumps, sometimes also as source code.

Hexdumps are the shortest and easiest format to type in. You must first enter the monitor:
CALL -151

Key in the hexdump exactly as it appears in the magazine, ignoring the four-digit checksum ($ and four digits) at the end of each line. When finished, return to BASIC with:
3D0G

BSAVE the program with the filename, address and length parameters given in the article.

Typing Source Code

The source code is printed to help explain a program's operation. To enter it, you need an "Assembler". Most of the source code in older issues is in S-C Assembler format. If you use a different assembler, you will have to translate portions of the source code into something your assembler will understand.

Computing checksums

Checksums are 4-digit hexadecimal numbers which tell if you typed a program correctly and help you locate any errors. There are two types of checksums: one created by the CHECKBIN program (for machine language programs) and the other created by the CHECKSOFT program (for BASIC programs). Both are on the "Starter Kit".

If your checksums do not match the published checksums then the line where the first checksum differs is incorrect.

CHECKSOFT instructions: Install Checksoft (BRUN CHECKSOFT) then LOAD your program. Press & to get the checksums. Correct the program line where the checksums first differ.

CHECKBIN instructions: Enter the monitor (CALL -151), install Checkbin at some out of the way place (BRUN CHECKBIN, A$6000), and then LOAD your program. Get the checksums by typing the Starting address, a period and the Ending address of the file followed by a **ctrl Y** .
SSSS.EEEE ctrl Y

Correct the lines where the checksums differ.

Writing to the RDEX editor

RDEX (are-decks) stands for: Reader's Data EXchange. We print what you write. When you send in articles, softkeys, APTs, etc., you are submitting them for *free* publication in this magazine. RDEX does *not* purchase submissions nor do we verify data submitted by readers. If you discover any errors, please let us know so that we may inform our other readers.

Remember that your letters or parts of them may be used in RDEX even if not addressed to the RDEX editor. Correspondence that gets published may be edited for clarity, grammar and space requirements.

Because of the great number of letters we receive and the ephemeral and unpredictable appearance of our volunteer staff, any response to your queries will appear only in RDEX, so it would be more appropriate for you to present technical questions to the readers and ask for their responses which will then be placed in the Apple-RDEX.

How to get a free library disk

Whenever possible, send everything on Apple format (5.25" - DOS/ProDOS or 3.5" - ProDOS) or IBM format (3.5") disks. Other formats are acceptable but there may be some delay as we look for someone to translate it for us. *(If you use a 5.25" disk, when we print your letter, we will return your disk with the current library disk copied onto it .)* Use whatever text editor you like, but tell us which one. Put a label on the disk with your name (or pseudonym) and address (if you want to receive mail). Don't reformat any programs or include them in the text of your letter. Send Applesoft programs as normal Applesoft files and machine language programs as normal binary files. We have programs to con-

vert them to the proper format for printing. If you are sending source code files, and you are not using the S-C Assembler, send them as normal text files.

When to include a printed letter

Don't include hardcopy (printout) unless:
a. You are writing about a bug or other printing error.
b. You are writing to ask for help.
c. You are answering another readers help request.
d. You are writing about your subscription or sending an order for back issues or software.

Bugs, requests for help and answers to requests for help are bumped to the head of the line and go in the very next issue. All other letters are printed in the order that we receive them.

Writing to get help

When writing to request help, be sure to include ALL relevent information. The more information you include, the easier it is to find a solution. There's an old saying that goes "A properly framed question includes 90% of the answer".

How to get mail

If you are interested in receiving mail from other readers, be sure that we have a current address. If you use a pen name and want to receive mail, we need to have your address. Our readers privacy is important, so we will not print your address unless you specifically say too.

How to write to RDEX authors

When writing to one of the RDEX authors. Write your letter and seal it in an envelope. Put your return address, the authors name (as it appears in RDEX) and *the correct postage* on the envelope. Put this envelope into another and send it to RDEX. We will put the correct address on your letter and mail it for you.

Help Line

These readers have volunteered their time to help you. Please call only within the given time frames (corrected for your time zone). No collect calls.

Jack Nissel (Disk Protection, 7-10PM EST) (215) 365-2905

Electronic Bulletin Board System (BBS)

Dave Goforth is the sysop for the Computist BBS. The number is: (206) 581-9292. If you already have a User ID# and password, sign-on using the User ID#. If you are a new user, it may take a day or so to validate your new ID# and password.

COMPUTIST

U.S. $3.75

Canada & Mexico $7

ISSUE 71

Features, Notes and such:

Table of Contents

Softkeys:

Bitkeys:

APTs:

Playing Tips:

IBM Softkeys:

IBM Playing Tips:

IBM APTs:

Charles R. Haight

Zip Chip vs Rocket Chip

It seems that Rocket chips are going to become a little scarce. The word we received is that Zip Technology won the lawsuit against Bits and Pieces Technology (BPT) and that BPT cannot manufacture any more speedup chips.

While the Rocket chip is deceased, the 8Mhz Zip Chip is shipping (we've already sent out a few) and we can send you one for $157 plus $2 for UPS (or $4 for Second Day Air, if you're in a hurry). Pricing is pretty predatory so you may be able to find a lower price from one of the mail order houses. Let us know if you do.

New Address

Computist is moving its offices to Eatonville (where I live). No more 60 mile round trips to Tacoma whenever work needs to be done on the magazine! That means I can work on Computist stuff in the evenings and be able to spend a little more time working (and a lot less time commuting).

Tighter format, more data

Back to the compressed format. So many of the letters that I've received say it's OK, that I'm trying it out. It's harder than it looks to use 4-columns per page, especially when there is a lot of source code. Text is easy, of course, since it is handled automatically by the page layout program. But commented code is too wide and when it wraps around, it's not very readable. That means extra editing must be done, which takes a lot of time. Also, some of your letters, have column type data that is formatted for 80 columns. That involves more time to convert to a narrow 4-column format. (Although, if it has to be in a wide multi-column format, I can put it in a box that spans several columns.) So, when you write to us, keep in mind the narrow columns. I really appreciate it.

Anyway, I'm getting used to this MAC and Pagemaker and it's getting easier to get each issue out each month. Our lateness on this issue is because of some extra time it took to work out the bugs in the 4-column format along with some minor problems with the mailing.

A Book for Issue 66

Speaking of formats, I've come up with a two column format for the Book. I'm trying it out with Computist #66 (the one that we are out of). The 34 tabloid size pages of text in #66 converts to about 65 pages in an 8 1/2 by 11 size book format. Anyone who orders issue #66, at regular back issue prices, will receive the book formatted pages. They're setup for 3 hole punch to fit in a three ring binder.

The COMPUTIST BBS
(206) 581-9292

The BBS is ready, but it's in the seedling stage and is going to require some careful nurturing. We don't have a clock card yet and we're wondering if anyone out there has an extra one that they could loan us (on a long term basis). Or, maybe we could trade for back issues or library disks.

Everyone who joined the Club should find a slip of paper in their issue. (I suggest you look for it, right now!!!) If you do not find the slip of paper and you have paid your club fee, contact us immediately. The paper is marked with two numbers and a word or phrase. The first number is your club number and should match the number on your label. If it doesn't, contact us immediately. The second number is your BBS user ID number. You should use this number when you sign-on to the BBS. The word or phrase is your initial password. After you sign-on, you may change your password to whatever you wish. We got the initial passwords by writing a program that randomly pulled words from a thesaurus program. So if your password is something weird, you know why. You wouldn't believe some of the words and phrases that came out of that thesaurus.

Hardware Happenings

The verdict's in on the Display card. Guilty of dullness and lack of interest. Out of about 4000 subscribers, only 15 were interested in assembling the card.

Here's why that isn't enough orders. The setup fee for manufacturing an average size card is between $200-300. Then, the actual card costs between $10 and $20 each. Since these are very small cards, the real cost is $7.60 for one and $6.90 for the other, or $15 for both.

If we share the setup fee, (300/15 = 20) that would mean that the cost to each of us would be $20. Now add the $15 for the 2 card set and $15 for the components and we have a grand total of $50.

That's pretty steep for a card that is essentially a toy. (And an unassembled toy, at that!)

So we returned the checks and didn't order any Display cards. The board layouts, parts lists and net lists are available for anyone who wants to make the card on their own. Send a letter to Karen, along with a dollar for postage. If you do make the boards yourself, consider making a couple extra and selling them to other interested readers. Or maybe someone knows where we can get the boards done for less?

Other Hardware?

Now the question is, "What do we do with the other projects?" We have two other projects that are at the board stage and a dozen more in various stages of progress.

The first project is a special memory card that holds 8 - 32K SRAMs (Static RAM) for a total of 256K of RAM. The card looks like a 256K disk drive, a very fast disk drive. A special Integrated Circuit (IC) from Maxim maintains the voltage to the SRAMs when you turn off your computer. It also detects a power failure and locks out writes to the card to protect the memory. You can save data to the card (just like to a disk) and turn off your computer. Then, in the morning, when you're ready to start computing, the data would still be there. I use the prototype to hold DOS 3.3, ProDOS 8 and my most often used utility programs, so I can switch between them in a fraction of a second. I have the card plugged into slot 7, so when I first turn on my IIe, DOS is instantly loaded. Naturally, I call it the "Instant On" card. The prototype cost almost $400 (2 blank cards plus the setup charge). And now that I've had time to play with it, I've thought of some changes that would be nice to make before the board layout is finalized.

$400 is a bit expensive. You could buy the Stat Disk card from A2 Central for a lot less. But, if we had 50 readers who were interested, the cost would come down some. It would be about $6 each for the setup fee (300/50 = 6), $25 per bare card (less if we don't want gold plating on the connector), $14 for misc. parts (some IC's and sockets), and $10 per 32K SRAM chip. The total cost of the board (stuffed with 256K of RAM) would be about $123. That compares favorably with the Stat Disk card with 128K for $209. (Although, the Stat Disk comes with some valuable software and is expandable beyond 256K.)

The second project is the "Smart" card (for the IIe). This card has its own microprocessor, and it can stop the IIe, and examine and change any location in memory, all without leaving any trace of it's activity. The perfect card for the active COMPUTIST. If you are playing a game, you can stop it, at any time, and give yourself some more men or weapons. Hackers can use it to do parallel processing, or program the card to do background sound effects while your main program is running in your IIe, all without the main program knowing what's happening. It also has a programmable break point monitor. That means that you could tell the card to freeze the Apple and come to you whenever a certain address is referenced by the program in memory. Krackers take note, we put the breakpoint monitor in there for you. It makes it real easy to track what a protected program is doing.

Dave Goforth, the COMPUTIST BBS sysop, asked about a RAM card for the IIgs. We'll need some physical measurements of the space around the IIgs RAM expansion slot but it doesn't look too tough. A look at the IIgs Hardware Manual shows that all the signals you need for the first 4 Mbytes are supplied by the expansion slot. I figure about $100 for the completed board (including gold plated connectors) and $100 for each 1Meg SIMM. So a 4 Meg RAM board would cost about $500.

Keep in mind that all of these costs are for unassembled boards, you have to do the soldering and assembly yourself. While soldering is not difficult, you do have to take care and have a steady hand. This also presupposes that enough readers are interested in a certain project to make it possible to order the boards. We're not set up here to become a manufacturing company, and if we did, we would probably have to charge a lot more for each board, just like other companies. They also have to recoup the cost of designing and testing their hardware. I'd rather donate my time and keep this on a one-on-one basis with interested readers so that the costs stay reasonable.

But we don't know if anyone is really interested. Maybe it's just that the first project was uninteresting or was poorly presented or if maybe there's not really enough interest in hardware anymore. Personally, I like designing new hardware. But, this is your newsletter and if you would rather not waste any more space on hardware, then we won't. So, be sure to write and let us know how you feel.

And if you are interested, please tell us what kind of project you want to see. That way, we won't be wasting time working on uninteresting projects. I'll take a look at what you want and tell you if I can design it, and if so, what I think it will cost to build it. Then we can go on from there. Is that a deal?

David Goforth   WA

Well I'm sorry to hear about those people who said that when their subscription ran out they would not be renewing because:

The COMPUTIST BBS is here!!!

Hopefully they know someone else who subscribes and will see just what they will be missing and reconsider. I too had thought as they did and did not like the tabloid format and briefly considered calling it quits, but I knew that I could not find the information that COMPUTIST prints in any other magazine and even sent in my $10 to

---

help them out AFTER the tabloid format started. I even helped send out issue #70 (two weeks after I'd received #69) and discovered exactly what they are going through. It's difficult to imagine but I will attempt to describe as a subscriber what I saw.

As Charles Haight has said, they have an all volunteer staff (so they can't cut costs there since there are no paid employees) and I was only one of six people working to get you and nearly 4,000 others their copy of COMPUTIST before Christmas. Second, the only money spent is for the actual printing and mailing costs of the magazine (the fewer the magazines, the more it costs per issue) we even put on all the address labels (which were printed on the computer in the office), sorted, and bundled them. About 8 hours non-stop work for everyone (I know, a days work is not bad but, they usually don't have that many people either). They had about another 2-4 hours work ahead before they would be completely ready to mail (Postal system is very picky) when we quit for the day. Then as I was about to leave we began discussing the upcoming issue (#71) which would have to be finished and ready to mail by the 12th of December and this was November 22nd (day before Thanksgiving)!

So, after seeing the working conditions, where the money goes, and the antiquated equipment being used (II+'s, Franklin ACE 1000's, a few IIe's and one Mac that was on loan from another subscriber) I think that Charles Haight and all the subscribers that have been with them since the early 80's deserve much more credit than we've been giving them. To keep this publication alive and going this long with what they have is incredible. But don't get me wrong, $32 a year IS outrageous but the ONLY way to reduce the cost of the magazine is to double the number of subscribers which would effectively have the following results:

1. Double the gross income of COMPUTIST while,

2. reducing the cost per issue for printing.

Somewhere, in there, is enough net profit to hire a full time editor (why? because as of Nov, there are letters dating back to June that have not yet been replied to because this volunteer staff have other jobs and families including Mr Haight - NO ONE GETS PAID and time is not something most people have a lot of unless they are getting paid), upgrade some of the equipment they are using AND allow for reducing the cost of the magazine to it's subscribers, it may even allow returning to the nice magazine format if that was the preference over reducing the $32 annual charge. I've found two subscribers by word of mouth in the last month and one about three months ago who liked it so much he bought EVERY back issue from day one thru his current which was around #64 (I couldn't afford that but HE found a way because he believed in what COMPUTIST stands for and didn't want to see it perish, besides he needed a lot of the older softkeys). No one person can do all of it, if each of us can get just two new subscribers or at least talk those considering quitting to continue we may have a real magazine at an affordable cost again... Now on to the BBS!!!

I have been a subscriber to COMPUTIST since issue 15 when I first heard about it while in Okinawa, Japan. I've been watching all the requests for a BBS and after setting up my own BBS (the GOPHER HOLE - (206) 584-3320) in September, modifying the software to suit my needs and to get a feel as to what was involved in running a BBS, I approached Mr Haight as a volunteer for the COMPUTIST BBS. I expected to use my IIgs and add an area for COMPUTIST subscribers to the existing board. He instead decided to have a dedicated system and supplied me with a IIe and all necessary hardware to set thing up and get it going. Well it's up and as of now this is what's available and how it works:

All subscribers who sent in their $10 (or more) already have full access granted to the system. (See the editorial. ed) Others will have to log on as new users with limited access until validated (usually 1-3 days). The following areas are available:

Bulletins: This is where informative information about COMPUTIST and the BBS is listed. Recommend all users read this the first time they log on. The Most Wanted list is also in this section.

Messages: Is broken down into three areas:

1) RDEX for inputting letters to COMPUTIST and allowing other subscribers to read and answer them possibly even before publication.

2) E-MAIL is for sending private letters to other COMPUTIST subscribers.

3) CLASSIFIED for placing ads just as you would in a newspaper.

To decrease the amount of time spent on the BBS and long distance rates by the phone company, each message area has the ability to do a 'F'ile transfer. If you have a fairly long article or many small ones and wish to send them as fast as possible, I suggest you do the following: Enter the RDEX message base and leave a short note that you wish to have the attached file published. Then enter a blank line and a menu will appear, press 'F'ile and then send the file using XMODEM protocol. After the transfer is complete then select 'S'ave. I recommend using 'ShrinkIt' or some other type of file compression method

---

beforehand to make the file as small as possible. This will make the transfer even quicker and will also allow multiple files to be combined in one transfer (NOTE: The 'F'ile command will ONLY allow one file to be attached to each message).

Files: Public. Domain software library and programs published in COMPUTIST. This area is divided into DOS 3.3, ProDOS, IIgs, and RDEX. The RDEX is for files that you wish to send to COMPUTIST for publication.

Utilities: A current User list and Personal data area is here to change your password and other information about yourself. ALL personal data is kept strictly confidential and not made available to other subscribers (except name).

Help: There is on-line help for all commands available on the BBS.

The BBS is using Applenet v2.1c (a modified version of 1.3c by Kevin D'Hayse). It supports 300 - 2400 baud and XMODEM protocols. To call, set your terminal program for;

8 data bits
1 stop bit
No parity
Echo Off

Baud to the highest baud rate (up to 2400) that your modem supports.

NOTE: Some long distance calls get a lot of noise on the line and become almost unusable at 2400, if this happens, changing to a slower baud rate (1200) usually works much better (you cannot do this while on-line, you must hangup and call back at the new baud setting).

The number is not a toll free call, all calls will be billed by the phone company to the callers number. The BBS will not accept charges for long distance calls so be sure that the person who pays the bill (if not yourself) is aware of your calls and the phone number you are calling. All questions about the BBS or it's use should be addressed to "SYSOP" if left on the board or David Goforth if mailed to COMPUTIST. The BBS will change and grow rapidly in the first few months as new areas and features are added. If there is something specific that you would like to see incorporated please let us know.

Many people think that the magazines should be made available on the BBS for downloading. I agree but what you will get will be: An all text version of the issue, graphics have been done with a Mac and are in such a format and resolution that not even the IIgs can display them. So for hardware projects you would not have any drawings or schematics to look at. Also Program listings would still be that, text listings, and the biggie; Can you call long distance, spend 30 plus minutes downloading (if the file is compressed), and print out the magazine (not counting having it bound) for less than $3.75 an issue? If so then please let us know your secret so we can save some money too.

All those who have the indexes (free ones) for all the COMPUTIST magazines that want to share it with others, please send it as soon as possible and I will incorporate it into the BBS. If at all possible (depending on the format - preferably Appleworks DataBase or text) I'll include a search routine for people to scan the index for specific topics, programs, etc.

One more word about this new volunteer, I own an Apple IIgs with the PC Transporter and can read the following disk formats; Apple 5.25 (140K), Apple 3.5 (800K), IBM 5.25 (360K), and IBM 3.5 (720K) so you can submit your inputs on any of the above disk formats for publication, hopefully this will allow for more IBM users to send inputs. All inputs should be submitted in standard ASCII text files if possible without any embedded special formatting codes and returns only at the end of paragraphs. This will greatly reduce the amount of time it takes to edit the submissions for publication. Also, any programs or hex listings should be sent in their original format (Basic and Binary files should not be converted to text). For more information of submissions please read the Readers Data EXchange printed at the beginning of every issue of COMPUTIST.

PS. Add one more subscriber. As I was writing this I 'chatted' (on the Gopher Hole BBS) with a caller who has been using an Apple IIe for over five years and never heard of COMPUTIST (and living in Tacoma, WA no less). He recently cancelled all other magazine subscriptions because of all the ads and very little information that they were publishing. He had no idea that this type of publication existed and really wants to see it. All it takes usually is to tell someone about COMPUTIST, then show it to them. Don't tell them the cost, let them see it for themselves when they see the magazine. Their reply is usually something like, "It's kind of high but there's no ads and you don't see this info anywhere else and I've subscribed to all Apple magazines in the past...". They usually end up joining and if you ask them about the cost you usually get something along the lines of, "No problem, I'll just drop these other two that I don't read anyway and even save a few bucks for something better...". You'd be surprised how easy it is, just tell and then show. Let the magazine speak for itself and the person decide. The only thing you have to watch out for is to be sure that you get YOUR copy back.

PPS. Oh, before I forget!! The Computist BBS phone number is (206) 581-9292. Hope to "see" you soon.

The PRODUCT MONITOR

RATINGS

☆☆☆☆☆	SUPERB
☆☆☆☆	EXCELLENT
☆☆☆	VERY GOOD
☆☆	GOOD
☆	FAIR
?	POOR
??	BAD
X	DEFECTIVE

Hopefully, your Christmas money piggy bank yet retains a few coins earmarked for entertainment software purchases. As promised last issue, this month's focus is upon several promising game releases. Next time, for sure (?), I shall put the wraps on all but end-of-year '89 funware and zero-in upon utilities.

Pool of Radiance
☆☆☆

$44.95 for 128K Apple II
Cluebook: $12.95

Beyond any question, 1989 produced absolutely the best crop ever of quality computer adventures; which, I hope, explains why so many good products— for example, SSI's Advanced D&D releases— are only just beginning to get the attention they deserve. In "Pool of Radiance" your party of up to eight heroes is launched upon a long-playing quest to deliver the city of New Phlan from the evil mechanations of a mysterious arch-criminal known only as "The Boss". To this end you undertake a series of City Council "commissions" to obtain key artifacts, treasures, lost lore, AND, sometimes, take on the task of clearing one of the monster-infested suburbs. With each success your party attracts increased notice from the Boss's stooges (and their minions) until, finally, you tackle the Boss himself.

Though you will, occasionally, find yourself on an Ultima-style map of the nearby Wastelands, most of "Pool"'s action is set in the city's "Civilized Area", a suburb, or underground, in a dungeon. Here, and in some special Wilderness locations, you get a rather bland 3-D perspective view offering just enough detail for identification of buildings, doorways, arches, etc.. (This game badly needs a hefty dose of IIgs super-res and sound.) Aside from helping to fill-in some thirty-odd maps, your explorations can yield treasure, weapons/armor caches, and valuable information, supplied via a reference to one of the Journal or Tavern Tales entries in the game's "Adventurer's Journal" booklet.

Explorations also produce confrontations. Upon an encounter, the program displays a picture of one of the encountees (e.g. a hulking giant) and, more often than not, allows at least a chance at avoiding conflict. Should hostilities ensue, your party plus the enemy's forces appear on a scrollable combat map and the REAL fun begins! While a "Quick" option permits speedy, satisfactory computer resolution of easy battles, getting to crunch (blast, slice, ...) monsters is, after all, one of the big payoffs for your labors— and "Pool'" offers one of the better tactical combat schemes around.

Since race, occupation, and attributes are very similar to those found in "Ultima", "Eternal Dagger", etc.— as, in fact, are the magic, weapons, and armor systems— this is a game you can expect to start and 'get into' with a minimum of hassle. With a scenario boasting mystery, humor, and the promise of ever more challenging adventure, getting out is not so easy! (Supplied on four double-sided diskettes, "Pool of Radiance" comes with manual, "'Journal', and decoder wheel— to which I strongly recommend adding the map-filled "Clue Book".)

The Duel
☆☆☆☆

$44.95 for 512K IIgs
The Supercars: $19.95
The Muscle Cars: 19.95
California Challenge: $19.95 (needs 768K)

The first time I tried Accolade's "Test Drive" ($34.95 for 128K Apple II) my reaction was something like "This is a darn good game"; and, then, "I bet that if they release a IIgs version, it'll be fantastic". Well, they did, and it is. "The Duel" comes with two cars: the Porsche 959 and Ferrari's F40. As in the first "Test Drive", you can check out a picture + specs display for each car; and, each car actually does offer its own distinctive 'feel'. " 'Super Cars" adds five hot sportsters to your stable (including the '89 Corvette ZR1 and,

my favorite, the '85 Lamborghini Contach 500S); while, in " 'Muscle Cars", you get five monster engine legends from the '60's (e.g. the 425 hp '69 Dodge Charger Daytona, the 550 hp COPO ZI-1 Camaro, etc.).

Race the clock or tackle a computer-controlled competitor, either way you select the cars AND the course. So far, Accolade offers two "scenery sets". The game diskette set takes you through forests, mountains, and deserts; whereas, "California Challenge" amounts to a north-south tour of the state's more spectacular sites (e.g. redwood forests, ocean coasts, etc.). Even at high speeds, each set is good for several minutes— much more if you get caught by the highway patrol. Settable skill level adjusts the frequency of hazards like oil slicks, rocks, and traffic. (When you're tooling along at 200 mph, ANYTHING on the highway is a hazard!)

With joystick control and your behind-the-wheel super-res view, "The Duel"'s smooth animation and realistic sound come about as close to putting you on the road in a four-wheeled rocket as is possible. Even the necessary memory updates are masked via periodic service stops, where, beside fuel, you get an evaluation of your driving performance. (Naturally, top scores for each course are maintained on disk.) Aside from a documentation error which specifies CTRL-J and CTRL-K for joystick/KB control selection (J and K are correct), and the apparent requirement (undocumented) that RAM disk be OFF, "The Duel" has proved a flawless performer, good for many hours of enjoyable road-burning.

The Usurper: Mines of Qyntarr
☆☆

$29.95, Includes 128K Apple II and 512K IBM versions

Sir Tech's new TEXT-only release sends you into the dreaded Mines to recover the Orb of Baellon— the latter being the fabled key to disposing of the evil Usurper. Encompassing well over 100 'rooms', "The Usurper'" is a Zork-like adventure offering prosaic descriptions, interesting places and personages, and several worthwhile puzzles. Parsing (i.e. how well the program understands user inputs) is several notches above Eamon grade; but, still, just adequate, considering the tasks at hand. Often, for instance, Usurper fails to recognize items mentioned in its descriptions; and, sometimes, misses obvious substitutions. (e.g. "Get Scrap" will not suffice to pick up a Scrap of Parchment.) Happily, once you adjust to the game's moderate smartness level, 'playing the parser' situations are minimized; and you can concentrate on the adventure. Supplied with an attractive illustrated manual, "The Usurper'" is cute, entertaining, and good for several evenings of play.

Fast Frames, Updates, Etc.

Wiz V: Kettles & False Mazes

Dr. Hackenbush's question in #68 provoked a look through my "Wizardry V" notes, and, very probably, has re-hooked me on finishing the quest. This IS one of the great adventures. As to where to find the Laughing Kettle: on Level I at coordinates 22 East, 23 North, there is a secret door in the north wall which leads to the Kettle. He offers some useful clues for prices ranging from $500 through $5000. (Incidentally, don't be fooled by his jolly appearance. If attacked, the Kettle proves to be tough, powerful magic user; AND, should you win, the temple healers will require that you pay for his resurrection!)

The false mazes encountered by some "Wiz V" players ARE very strange places (places, in fact, where you are not supposed to be); but the phenomenon is not the result of some bug. (In the first batch of "Wizardry V" packages shipped, some did have bad diskettes— quickly replaced by Sir Tech— but, these simply crashed the game.) Rather, the problem stems from player errors which go undetected because 'correct diskette' verification is limited, mainly, to occasions when you move from one level to another. For example, if you take an elevator to Level 3, the program will insist that Level 3's diskette be in place before play continues. If, once on Level 3, you should swap in Level 5's diskette, the program will try to construct a maze from (and do updates to!) whatever it finds. Naturally, the Level 5 'player copy' diskette is ruined; and you find yourself in an oddly 'inside-out' maze which can look just genuine enough to occupy your best mapping skills for an hour or so.

My excursion into a false maze began when, using Roger Wagner's "Soft Switch" (on a IIgs), I decided to jump back to an earlier position on another level and forgot to swap in the new level's diskette. Well, the program had no way of knowing that some sly user was playing fast and loose with the adventure. It went merrily along, reading from and WRITING TO the other level's diskette; I ended up mapping a hefty chunk of "Wizardry V"'s 'twilight zone', and the diskette side was bombed. A reasonably satisfactory cure is to get out your set of originals and recopy the side that's been damaged. Everything on the restored level will be reset— you may have to reopen a few locks, etc.— but, at least, the quest can continue.

Fast Break ☆

To work, a basketball simulation must be able to take the necessarily limited inputs from a user and 'fill-in the gaps'. When it succeeds, as in Activision's "Two-on-Two", you discover that control is easy and your man looks and acts like a basketball player. When it fails, as in Accolade's "Fast Break" ($44.95, for 768K IIgs), you have a gang of impossible-to-control stumble bums.

Tomahawk ☆☆☆

I suppose that, as long as I'm going to repeatedly refer to "Tomahawk performance" in joystick reviews, it's a good idea to say something about the best flight sim program available to IIgs users. Basically, the game puts you in the cockpit of an AH-64A Apache attack helicopter (code name: "Tomahawk") armed with a 30 mm automatic cannon, 38 rockets, and 8 'smart' Hellfire missiles. The super-res display presents a through-the-windshield forward view, an optional war map insert, and high-tech instrument cluster (comprehensive ship status displays and clickable controls plus locator and targeting aids). Very nice; and, when you throttle-up, you'll hear the whine of your engines and thuk-thuk-thuk of the rotors in (optional) stereo!

Missions range from "Training" through three increasingly difficult combat scenarios. To up the challenge level, you can select a higher pilot rating and/or toggle such options as "Night", "Cross winds", and "Cloudy" (with a user-settable cloud base down to 50 feet). Thanks to excellent instrumentation, you COULD fly your Tomahawk through a cavern at midnight— fortunately! While the 3-D line drawings (of buildings, mountains, tanks, enemy choppers, etc.) provide reliable visual distance cues, when you're over bare ground only the altimeter supplies any indication that you are about to plow into the turf.

Using joystick + mouse controls, Tomahawk is easy to fly; but flying well (i.e. landing and combat) definitely takes practice. Indeed, since you are flying without benefit of critical body position and middle-ear sensory cues, mastering DataSoft's Tomahawk probably means you're ready for a shot at the real thing! (I wonder if the Army would be interested in a little experiment.) Whether you seriously take on a combat mission or just enjoy zipping around and blowing up enemy targets, "Tomahawk" is first rate flight simulation fun. ($29.95 for 512K IIgs)

Deathlord ☆☆

When the Ninth Emperor commands you to root-out the "source of evil" threatening all Lorn, your six heroes begin a large-scale quest spanning sixteen continents packed with towns, ruins, dungeons, etc.. Looking and playing much like "Ultima III" 'done in Japanese', "Deathlord" locates all action on one its numerous maps (i.e. no forward-view mazes, monster close-ups, etc.). Character differentiation is exceptional— you have lots of races, 'occupations', and attributes to adjust— and the 84-spell magic system offers a nice range of powers. The scheme for combat, however, is about as simple as any non-tactical setup you are likely to encounter.

"Deathlord" delivers a crisp, attractive hires display and a scenario loaded with mini-challenges. (The Electronic Arts 200-plus-hours playing time estimate is probably not an exaggeration.) On the debit side, sound is barely adequate, fighters must choose from a small range of (poorly documented) weapons and armor, and all of the key designations (race, class, weapon, armor, and spell names) are in Japanese. While the latter feature adds a touch of authenticity; it would be would work better (i.e be more fun and less bother) if the Japanese terms were accompanied by direct translations.

Possibly, the game's most serious weakness is the failure to match new character abilities to level of challenge. Since things are set up to present a stiff test for any developed characters you may transfer (from "Ultima III", "Wizardry I-III", or "Bard's Tale I"), "Deathlord" tends to chomp beginning heroes like peanuts. Another flaw, placing character creation and party formation in a separate utility (i.e. you must re-boot), makes 'learning the ropes' all the more cumbersome. "Getting into" "Deathlord" definitely demands more raw effort than it should. Too much? If you especially enjoy big map, long-play 'old Ultima' adventuring, probably not. ($39.95, for 64K Apple II)

Becoming a Deathlord Survivor

Possibly, you are enmeshed in "Deathlord" (or, eventually, will be) and are perplexed by 'hero die-off'. (Even Gorbash, winner of Ultima's IV and V, M&M I and II, etc., put "Deathlord" on the back burner for a few months.) One solution, as usual, is 'super-natural intervention'— i.e. you diddle a few character parms and, SHAZAM!, your guys suddenly feel stronger, faster, smarter, and richer! A second trick is to minimize the damage should a 'worst case scenario' eventuate.

First, the SHAZAM solution: Using Central Point's "Copy II Plus" ($39.95, for 128K Apple) and DOS alteration data found in Rob Fiduccia's "Deathlord" article in issue #63, we scanned the Scenario A diskette and found character data in sector $0D on track $13. (1. Go to 5.25" BitCopy. 2. In the Sector Editor enter "P" to get to the DOS patcher and then "Custom". 3. Under "Address",

change D5 AA 96 to D5 AA D6 and DE AA to DE B7. Under "Data", change D5 AA AD to D5 AE AD and DE AA ... to EA AA ... . 4. Exit the patcher and Read T/S $13/$0D. NOTE: Should you wish to scan the diskette starting at track $0, expect to encounter a few unreadable sectors along the way. Just skip these and continue.)

On the Sector Editor screen, you should find character data arranged in six-byte groups (one byte per character) as follows:

$42:	Hit Points, max (low byte)
$48:	Hit Points, max (high byte)
$4E:	Hit Points, current (low byte)
$54:	Hit Points, current (high byte)
$66:	Level
$72:	Strength
$78:	Constitution
$7E:	Size
$84:	Intelligence
$8A:	Dexterity
$90:	Charisma
$96:	Power, current
$9C:	Power, max
$CC:	Age
$D2:	Food
$D8:	Gold (low byte)
$DE:	Gold (high byte).

Edit as desired, write your SHAZAMed characters back to disk, and your quest against the Deathlord is a 'whole new ball game'.

Granted, if you max-out every parm and give each character $10,000 gold, there isn't much your party can't handle. If, however, you merely go for a "fair start", then the inconvenience associated with having several characters wiped out remains a threat. You can make a character backup each time you start the game; but, of course, you won't. It's too much bother, the backup won't restore hours of progress made in the current session, and, besides, there's a better way! Use the "Make Scenario" function to create a copy of each current scenario diskette on its unused side. During play, you just do an occasional SAVE on alternate sides. If your party is wiped out and saved as dead on Side One, you can restart from Side Two, etc.. Whatever happens, you will never lose more than the progress made since the last opposite-side SAVE.

Vendors

Accolade: 20813 Stevens Creek Blvd., Cupertino, CA 95014 (408-446-5757)

Central Point Software: 15220 N.W. Greenbier Parkway #200, Beaverton, OR 97006-9937 (503-690-8090)

Electronic Arts: 1820 Gateway Drive, San Mateo, CA 94404 (415-571-7171)

Intellicreations/Datasoft: 19808 Nordhoff Place, Chatsworth, CA 91311 (818-886-5922)

Sir-Tech: P.O. Box 245, Charlestown Mall, Ogdensburg, NY 13669 (315-393-6633)

Strategic Simulations Inc.: 1046 North Rengstorff Ave., Mountain View, CA 94043 (415-964-1353)

Softkey for...

Soko Ban
Spectrum Holobyte

Requirements:

Apple IIe or IIc (128K)

Jim Hart's softkey for Tetris (COMPUTIST 62, page 23) may have been a "half-key" for Tetris, but it is a full-key for Soko Ban. Soko Ban is a strategy game in double-hi-res of a warehouseman trying to move boxes through a maze to the loading dock. There are 50 different mazes on the disk, so it should provide hours of entertainment (or frustration).

The Soko ban disk is normally formatted, except that the epilogues have been changed to FF FF instead of DE AA. Once this has been corrected, Jim Hart's Tetris "half-key" can be used, except that the code is in a different location. Jim saved me many hours of work by publishing the bytes that he found on the Tetris disk. Many softkey contributors do not send in a listing of the code that needs to be changed, rendering that softkey useless for slight alterations in the copy protections.

1. Boot your DOS 3.3 system disk.

2. Tell DOS to ignore checksum and epilog errors and use COPYA to copy both sides of the Soko Ban disk.

POKE 47426,24

RUN COPYA

3. Make the following sector edits to the copy.

Trk	Sct	Byte(s)	From	To
00	05	0C-29	A9 0A 85 FC A6	A9 E7 85 F8 85
			2B BD 89 C0 BD	FB A9 FC 85 F9
			8E C0 A9 80 85	85 FC 85 FF A9
			FD C6 FD F0 71	EE 85 FA 85 FD
			20 AF F0 B0 6C	85 FE A6 2B BD
			A5 F9 C9 00 D0	89 C0 18 90 52

I read your editorial on your financial problems, and have two suggestions. 1. Raise the rates. I don't think $5 per issue is unreasonable. 2. Try to get your mag into computer stores and software stores. This would get store sales plus increased recognition of a unique publication and lead to more subscriptions.

Raising the rates might seem like a solution, but it's not and I only brought it up in order to eliminate it. If we raised the rates, more readers would not renew and the rates would have to go up again, and more readers would not renew and the rates would go up, and more... A vicious cycle and not one that we want to get caught up in.

As to computer stores, we try that one on a regular schedule but the results are not encouraging. Most stores do not buy magazines direct. They use a periodical service that keeps their racks loaded. These periodical services do not pay up-front, instead, they pay up to 120 days later. And they like to order a lot of issues but they claim that they don't sell all of them and that they had to burn/destroy most of the copies (about 90%). One periodical service sent us affidavits claiming to have destroyed more copies than we had sent them. So you can see that it's a lot more involved than it seems and there are a lot of rats out there. Some smaller stores do order direct, but they don't make a lot on magazine sales and they don't like you to see how low the mail order prices are and they don't want you to know how to backup your software and so on... Another dead end, but we keep trying and we do have a few regular dealers, mostly in foreign countries. .RDEXed

George Sabeh — PA

I would like to share the following softkeys with your readers.

Softkey for...

LA Crackdown
Epyx

Requirements:
COPYA
Sector editor (such as Copy II Plus)
2 blank disks

1. Boot your DOS 3.3 system disk.

2. Tell DOS to ignore checksum and epilog errors and use COPYA to copy the disk.
POKE 47426,24
RUN COPYA

3. Make the following sector edits to the copy.

Trk	Sct	Byte(s)	From	To
00	05	10	A6 2B BD 89 C0	A9 E7 85 F8 A9
			BD 8E C0 A9 80	FC 85 F9 85 FF
			85 FD C6 FD F0	A9 EE 85 FA 85
			71 20 AF F0 B0	FD 85 FE A9 F3
			6C A5 F9 C9 0F	85 FB A9 70 85
			D0 F1 A0	FC D0 50

4. Write the sector back to disk.

5. Copy side 2 with any copy program. It is in normal format.

Softkey for...

Into the Eagles Nest
Mindscape

Requirements:
COPYA
Sector editor
One blank disk

The softkey in Computist 66, p23, by Zorro was of great help. My copy had altered epilogs and required further modifications.

1. Boot your DOS 3.3 system disk.

2. Tell DOS to ignore checksum and epilog errors and use COPYA to copy the disk.
POKE 47426,24
RUN COPYA

3. Make the following sector edits to the copy.

Trk	Sct	Byte(s)	From	To
00	07	0C	38	18
00	08	63-65	BD 88 C0	EA EA 60

4. Write the sector back to disk.

Softkey for...

Jump Jet
Avant Garde

Mission Escape
Microsparc

These two programs use modified epilogs and can be copied by poking 47426,24 to ignore epilog errors and running COPYA to produce a working copy.

1. Boot your DOS 3.3 system disk.

2. Tell DOS to ignore checksum and epilog errors and use COPYA to copy the disk.
POKE 47426,24
RUN COPYA

Softkey for...

Tag Team Wrestling
Data East

Requirements:
COPYA
Sector editor

1. Boot your DOS 3.3 system disk.

---

2. Tell DOS to ignore checksum and epilog errors and use COPYA to copy the disk.
POKE 47426,24
RUN COPYA

3. Make the following sector edits to the copy.

Trk	Sct	Byte(s)	From	To
00	05	4B-4C	D0 F4	EA EA
		54-55	D0 36	EA EA
		5D-5E	D0 2D	EA EA
		70-71	D0 F4	EA EA
		7B-7C	D0 0F	EA EA

4. Write the sector back to disk.

Softkey for...

Cutthroat
Infocom.

This program can be copied by following the softkey for Sorceror (Infocom) in Computist #51, page 35 by Jack Nissel.

Softkey for...

Cavern Creatures
Datamost

Congo
Sentient

These two programs require a way to reset into the monitor. I used the Senior PROM. They can be copied using the Newswap Controller or similar copy program. I used COPYB.

Cavern Creatures:

1. Initialize a blank disk with a fast DOS.

2. Boot the original and reset into the monitor.

3. Capture the RWTS and copy using the Newswap Controller.

Congo:

1. Initialize a blank disk with "BOOT" for the hello filename.
INIT BOOT

2. Boot your original and reset into the monitor.

3. Capture the RWTS and copy using the newswap controller.

I would like to comment about the new format. Although I realize the need for change to save Computist and would be willing to accept any change to cut expenses, but I would much rather have the old smaller format. I would be happy with loose leaf binder type pages if they are smaller since they can be filed easily with the old issues. Also, I would like to see you publish a small book about different protection methods and ways to bypass it. It should be in-depth with comprehensive explanations. I am sure several of your contributing readers are capable of writing such a book and I feel most of the other readers will purchase it.

Captain Video — PA

I am writing in response to your questionnaire in issue 66 of Computist.

First, I would like to commend you for a fantastic magazine. I have been into computers for about three years, and I have learned more about my Apple GS from Computist than any other source. I feel it is well worth the $32.00 subscription price.

1) I would love to get Computist every two weeks with the tabloid form. I would be willing to pay the extra money for the increased subscriptions.

· 2) I think the Beginner's Notebook is a great idea. This would help to keep new subscribers from being overwhelmed by the articles in Computist. It would help keep them interested, which would result in their renewing their subscriptions.

3) I like the idea of bounties for stubborn programs that are on the Most Wanted list.

4) I think the free software giveaway is a good idea to motivate readers to write better articles. I don't believe there should be categories, just the three or four best articles in each issue.

5) The help line is also a great idea, but a lot of work for the volunteers. I could think of a lot of questions to ask.

6) The reader's Wish List is also a great idea to motivate readers to find new subscribers. The point system is a great way to run the program, but more work for everyone at your end. All of the above ideas are great, but I don't know how you are going to find enough time to do it all.

7) I do not think lowering the rates will increase the number of subscribers by any substantial number.

8) I don't mind the tabloid form for Computist, just as long as they keep coming.

9) I would accept raising the rates to $48.00. I prefer this to cutting the number of issues to eight per year, although I would be willing to accept this, also. My first vote is to double the number of yearly issues with the tabloid form to justify the price increase.

Although Computist is by far the best Apple magazine on the market, I believe the above changes will make it even better. Whatever has to be done, I'm sticking with you. I believe the money I've spent on Computist has been the the most wisely spent money that I have put into computing.

I would also like to send in my first softkey for the Apple GS.

Softkey for...

Impossible Mission II GS
Epyx

Requirements:
512K Apple II GS
3.5" disk copier
3.5" disk editor

The copy protection is a nibble count on tracks 20 and 21. Copying the disk will produce no errors or bad blocks. Booting the copy produces the message to insert the master disk, a sign of a nibble count. Use your 3.5" disk editor to scan for the hex code A2 20 A0 01. The code was on block $04A3 on my disk. At the very beginning of block $04A3 I found this:

```
01:    D0 64        BNE    1167
03:    20 60 2D      JSR    2D60
```

The code for the nibble count begins on byte 46. To break this protection, change the JSR instruction (20) to a LDA instruction (AD). This will stop the nibble count check and the disk will be deprotected.

1. Copy the disk.

2. Scan disk for A2 20 A0 01.

3. Locate the JSR routine 20 60 2D. (Mine was on block $043A, beginning with byte 03.)

4. Make the following edit:

Blk	Byte(s)	From	To
4A3	03	20	AD

5. Write the edit back to disk. That's it!

Brian A. Troha — WI

There are a few things I would like to comment on:

First, should COMPUTIST include other machines? Yes, only if it will ensure the continuation of the magazine. It's funny, I used to think if it wasn't an Apple IIgs program I didn't want to know about it. Then friends kept asking me to krack all these IIe programs for them. Now I find I like kracking any Apple software. If I had a Mac or an Amiga (or even an IBM, gasp!), I'd get into kracking programs for those machines. Right now I don't have either, but I still like to read about the copy protection schemes. Some day it may come in handy.

Second, the tabloid format: I hate it!, but if you need to use it to continue the magazine, then I'm for it. The idea of the year-end book style of previous issue is great. Think of it as a Tome of Deprotection '89 and etc. for the following years. Wouldn't it be nice to pick up a book (about the size of a small dictionary) and flip through a years worth of articles and find what your looking for?

Third, fake softkeys! This really bothers me! If you can't come up with something for real don't fool us with trash. As the RDeX prints what is sent in, they/we are forced to trust our writers, so let's not mess that up.

Fourth, I really don't like the idea of someone sending in 70 or so patches saying "here's my patch collection". I have a list of patches with around 95 programs on it. These are the most complete and accurate patches anywhere. Most are my work, however there are some from other authors or, at least, from their ground work. You could have nearly the SAME list as I have if you go through past issues of COMPUTIST and compile the patches, it's not really that difficult to do. Sure that's nice and all, but every author that worked on the articles that the patches came from deserve credit.

As an example: For some of my softkeys, it will take me anywhere from 5 minutes to an hour to krack and test a program. It can take several hours or days to try to tackle programs like Tomahawk, Hunt For Red October, and Dungeon Master. Then it's another hour (at least) to list the copy protection, type it in and add comments, and finish the article over all. Then get them together and send them in for others to read and learn from. I'm not getting rich doing this and that's not my intention, I want to teach others how to krack. Just like the late great ones: Disk Jockey, Reset Vector, Parity Error, etc. The point is: Credit is MY payment and nobody likes to get ripped off!

Softkey or APT: Well it's time to redefine Softkey (which was defined before the GS) as something like the following:

Softkey is a term which we coined to describe a procedure that removes, or at least circumvents, ANY copy-protection on a particular disk. Once a softkey procedure has been performed, the resulting backup copy can be copied by normal copy programs (COPYA, Fastcopy.system, etc). Such copies will no longer require a Keydisk or

---

the entry of a Keyword to run.

And now for a new word:

Hardkey is a term which was coined to describe a procedure that allows a non hard disk compatible program to be moved to/run from a RAM or hard disk. (examples: Mean 18 in COMPUTIST #68 & Bubble Ghost in COMPUTIST #71)

To Derek: Black Bag was NOT a user group, it was a pirate group now since disbanded. They had some talented people, I think they were the first to krack and release Wings of Fury in the pirate world.

I recommend the following two block editors for people who wish to start kracking software on the Apple IIgs:

The one in Copy][+: Just fine for making known patches and searching the entire disk for bytes.

Block Warden from ProSEL: This editor is very good, it has a follow file feature (as CP routines are usually in the 16 bit program file). The disassembly will follow the REP & SEP commands, that is the changing from 8 bit to 16 bit Accumulator and index registers. ProDOS 8 and ProDOS 16 calls are listed properly and Block Warden tells you which call it is. The block editor is worth the money for the whole ProSEL package by itself. Then you get the rest of the programs with ProSEL, an awesome bargain!

I used to use Copy][+ because I knew how to do everything manually, like finding the Key Block of files and such. Now I use Block Warden, as it's much more powerful and automated.

Warning: I have decided to write the following for everyone's benefit. If I make a comment about the way individuals do things, PLEASE do not take it as an attack. I'm trying to explain the why and why nots. Also, I'm NOT saying "I'm better", just passing along a few lessons and some information.

Let me begin with "encrypted code" (see Dan Halfwit's King of Chicago, Computist #68, page 13). When 16-bit system files are written (the $B3 or S16 file types) they contain segments or pieces of programs "linked" (or spliced) together. These files are usually written as relocatable due to the varying machine setups, that is, the amount of memory and number of CDA/NDAs installed. So, in memory, a JSL 034567 could show up as ANYTHING on the disk or just a plain JSL 000000 (Sierra comes to mind, right?). So when you make an absolute reference it might not work with every machine.

I found this out when I went to deprotect Draw Plus (Activision) and the program kept bombing. I overwrote a small section of code with EA's and, when it loaded in the ProDOS 16 loader, made the changes to the relocatable instructions. This is the reason you see so many GS kracks changing JSL (22) to LDA (AF) and JSR (20) to LDA (AD). The idea is to keep the instruction the same length and not care if the absolute address changes. We don't mind if the program LDA 0000, LDA FFFF, or anywhere in between as long as it doesn't go through the protection routine. Again, I'm not saying Dan is totally wrong, in fact he is right on the money. The only problem is he goes about the edits as though the GS programs are like IIe programs.

Now I would like to propose a the following procedure (that I use):

1. Figure out what kind of copy protection is used

2. Find the copy protection routines

3. Look for flags or special memory references

4. Kill the calls to the copy protection

5. Force the flag checks to pass

This reminds me of the time I used to go to summer camp. We played two team styled games: Capture the Flag & Spy vs Spy

Capture the Flag: You take two flags (and two teams) and set them a reasonable distance apart. You split your team into two groups: 1. Team A: Finds the other flag, grabs it and tries to run it back to it's own game area. 2. Team B: Tag the opponents Team A members who must now give up the flag or freeze for a period of time and most important guard their own flag.

Spy vs Spy: Take a same piece of paper with a point value written on. Hide it on your person. Try to run to your headquarters without being tagged by a member of the other team. If you are tagged the other team is allowed to search through your clothes (to a reasonable limit) to find the paper. This, by the way, is one of the best co-ed contact games I know of!

So why tell you about these games? Well we're all playing a combination of the two. Our team: Computist readers/members/RDEX and "The Other Guys": Software protectors. As you can see we have the numbers and the combined skill to beat them EVERY time. Think about it: If you cannot krack a program, chances are one of your 3800 or so team members can! This is our greatest strength. As an example:

Softkey for...

King of Chicago

Let's work with King of Chicago (Although I no longer have access to copies I'll tell how I kracked it). We know they have a flag somewhere in their code "game area." Our job: Find it, and bring it back (thus taking it out of the pro-

---

gram). Guarding the flag are several layers of Team B. As we work our way through we notice KC uses the nibble counting routine. They also use a mask so we cannot just search for A2 20 A0 01.

Dan tells us they use A9 20 00 & A9 01 00, so he is passing the captured flag to another team member. After looking through the code, they use two flags called 0022 & 0024 (as I remember) on the disk. So lets look for some references to that, searching the disk you will find a routine that compares those locations to the bench mark values used for the nibble count. However, in this routine, they also use a flag call 0026 and the routine that follows checks 0026 for a zero value.

The routine that checks 0026 looks like this, LDA 0026, BEQ +38 (or AD 26 00 F0 38). If 0026 is zero the code simply returns and runs the REAL KC game. If 0026 is anything but zero the code falls through to a routine that does a "playable demo" that ends with a randomly timed bomb! I traced these two routines back to the call and found JSL 010154 compares the nibble bench marks and JSL 010194 checks to see if 0026 is equal to zero.

I disabled the JSL 010154 and changed the LDA 0026, BEQ +38 to STZ 0026 BRA +38 (we just put up a fake "pass flag"). Although I don't have the exact block/byte changes, search your disk for 22 54 01 01 and change it to AF 54 01 01. Second, search for AD 26 00 F0 and change it to 9C 26 00 80. You now have kracked King of Chicago and captured a well hidden flag! Two points for our team.

Other programs that use the "hidden flag" approach are Xenocide (Micro Revelations), Final Assault & Impossible Mission II (Epyx). Taito also uses the flag approach on it's GS and IIe titles. I have been known to cheat by killing the calls to the copy protection, but writing passing values in to the flag location directly on the disk! Then I let the program read the implanted values, thinking it actual got them off the disk or from copy protection routines. Actually this is not cheating, it's just being as tricky as those who design copy protection. If I were to design a routine, I would add a flag called NO_EASY_KRACK in the middle of the disk read just to make sure the routine was run. Later into the game, I would check the flag and bomb out if not set right. Like Xenocide, King of Chicago, Downhill Challenge, Final Assault, etc, etc.

Softkey for...

Bubble Ghost

Accolade

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" Disk editor

To remove the annoying "What level is this?" question which is Keyword (pirate) protection follow these easy steps:

1. Make a copy of the Bubble Ghost game disk
2. Make the following edit to the copy only. (Block Warden [F]ollow /Bubble/ Bubble.sys16, rel block 78, bytes $9B74, $9B77, and $9BA2 for the changes)

Blk	Byte(s)	From	To
$17C	$174	AD 00 00	9C 00 00
	$177	F0 2E	80 2E
	$1A2	22 00 00 00	AF 00 00 00

3. Write the block back to the copy
4. Play off the newly deprotected copy

Once again, the whole requestor screen/routine is totally bypassed! However, the program was not hard disk compatible! I then set out fix this little problem.

Putting Bubble Ghost on Harddisk

The HardKey:

All files were being loaded like this: /Bubble/ WAVE. Which means, find the disk called "Bubble" and then load the file called "WAVE". I thought I could use the ProDOS 16 "1/" to make the program accept the current device. I now had to change all the "/Bubble" to something, so I came up with "1/Ghost" so Bubble Ghost will load ALL needed files off the current volume in the subdirectory GHOST. Use a copy of Bubble Ghost for the following steps:

Make the edit from "/Bubble" (2F 42 75 62 62 6C 65 in hex) to "1/Ghost" (31 2F 47 68 6F 73 74 in hex) at ALL the following locations: (If using Block Warden select [F]ollow /BUBBLE/ BUBBLE.SYS16)

Any block editor		Block Warden	
Block	Byte	Rel Block	Rel Bytes
$129	$9C	47	$5C9C
	$B8	.	$5CB8
	$CA	.	$5CCA
$171	$0F	67	$840F
	$2B	.	$842B
	$87	.	$8487
	$9F	.	$849F
	$B7	.	$84B7
	$CF	.	$84CF
	$E7	.	$84E7
	$FF	.	$84FF

| $172 | $143 | 68 | $8743 |
| | $158 | | $8758 |

Now get out a blank disk and format it with a name something like BUBBLE.GHOST or whatever. At this point I DELETED all unneeded files like the FINDER.DATA files, P8 in the system subdirectory, and FINDER.ICONS in the icons subdirectory.

Copy the PRODOS file and the SYSTEM and ICONS subdirectories to the new disk.

Copy BUBBLE.SYS16 to the new disk.
CREATE a new subdirectory called GHOST on the new disk.

Copy the WAVE and GHOST.SCR files into the new GHOST subdirectory.

Copy the DESSINS and SONS subdirectories into the new GHOST subdirectory.

A tree directory (other than PRODOS, SYSTEM and ICONS) would look something like this:

File	Type
BUBBLE.SYS16	S16
/GHOST	DIR
WAVE	BIN
GHOST.SCR	TXT
/DESSINS	DIR
GHOSTGS.CGS	S16
GS1.CGS	S16
GS2.CGS	S16
GS3.CGS	S16
GS4.CGS	S16
GS5.CGS	S16
GS6.CGS	S16
/SONS	DIR
LESSONS	BIN

Now, to upload Bubble Ghost to your hard drive, use COPY II+, the finder, or other file by file copier and copy BUBBLE.SYS16 and GHOST. The copy program will copy all the files and subdirectories in GHOST. Then copy the GENE.ICONS (from the ICONS subdirectory off the old disk) to your volume/ICONS subdirectory. To play the game simply launch BUBBLE.SYS16

Softkey for...

Battle Chess

Interplay

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

Battle Chess is great program to watch and kind-of easy to beat (which makes it fun!). However, you have to enter a move from a list at the end on the manual to start the first game. As I absolutely hate this kind of copy protection I decided to remove it. I simply loaded the system file (CHESS.SYSTEM) and followed it through to track down the call. I eventually found the routine and killed the call to it. I overwrote the call to CP routine with a STZ B908 which seemed to be some kind of flag. Notice: You can make absolute address references with this program because it runs under ProDOS 8 and is NOT relocatable.

To remove the Keyword (pirate) protection form Battle Chess follow these steps.

1. Make a copy of the Battle Chess game disk
2. Make the following edits to the copy only

Blk	Byte(s)	From	To
$3D8	$FF	20 DE AA	9C 08 B9

3. Write the block back to the copy

Softkey for...

Bubble Bobble

Taito

Requirements:
48K Apple II
5.25" disk copier
5.25" sector editor

The Apple II version of Bubble Bobble is a very poor port over by Taito/Nova Logic. I hope the GS version is a decent conversion. Never the less, the program is copy protected and I was able to remove the CP. After tracing through I found the CP routine is at $990A and all you have to do is stop it from ever running. I'm not really sure what it's looking for (it's like Arkanoid, but different than Qix) I was just interested in defeating it. So, the step by step would look like this:

1. Make a copy of both sides of Bubble Bobble
2. Make the follow edits to side 1 on the copy

Trk	Sct	Byte(s)	From	To
$04	$0D	$07	20 0A 99	EA EA 18
$14	$01	$07	20 0A 99	EA EA 18

3. Write the sectors back to the disk.
4. Store the original & play off the copy.

Softkey for...

ChessMaster 2100

The Software Toolworks

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

To remove the KEYWORD (pirate) protection from ChessMaster 2100 follow these steps:

---

1. Make a copy of the ChessMater disk
2. Make the following edits to disk one:
   For version 1.01 on disk one:

Blk	Byte(s)	From	To
$4E5	$2C	22 00 00 00	AF 00 00 00
	$61	D0 08	EA EA
	$C5	22 00 00 00	AF 00 00 00
	$118	D0 08	EA EA
	$185	22 00 00 00	AF 00 00 00
	$1D8	D0 08	EA EA

For version 1.1 on disk one:

Blk	Byte(s)	From	To
$304	$1EB	22 00 00 00	AF 00 00 00
$305	$20	D0 08	EA EA
	$84	22 00 00 00	AF 00 00 00
	$D7	D0 08	EA EA
	$144	22 00 00 00	AF 00 00 00
	$197	D0 08	EA EA

3. Write the blocks back to the copy

Softkey for ...

Crystal Quest

?

Requirements:
512k Apple IIgs
3.5" disk copier
Block Warden from ProSEL or other block editor with "follow file" command.

First a friend asked me to remove the "Is this a legal copy" question. Then he challenged me to do it in under 15 minutes. He also said "It sure would be nice to erase the high scores too!!" So I took him up on it and here is the results: I was able to remove the question, restructure the file layout, and reset the high scores. Follow the step by step instructions:

1. Make a COPY of Crystal Quest. (Make changes to the copy ONLY.)

2. RENAME the following files:
   CRYSTAL.SOUNDS to SOUNDS (The subdirectory of the sound files)
   CRYSTAL.SHAPES to SHAPES (A BIN file of the shapes)
   CRYSTAL.SAVE to SAVE (A BIN file of a saved game)
   MAIN.PIC to CRYS.PIC (The title pic)

3. CREATE a new subdirectory called "CRYSTAL"

4. Copy the files SHAPES and SAVE and the subdirectory SOUNDS into the new subdirectory CRYSTAL

5. Get into Block Warden and follow the file CRYSTAL.SYS16

6. Make the following edits:

Rel Blk	Rel byte	from	to	reason
7	C69	4D 41 49 4C	43 52 59 53	Chg MAIN to CRYS
7	C7B	2E	2F	Chg the "." to "/"
7	C8C	2E	2F	Chg the "." to "/"
47	5D46	20 7A 72 B0 0A	AD 7A 72 EA 18	Kill the question
63	7DFB	2E	2F	Chg the "." to "/"

Note: The changing of the "." to "/" tells the system file to look for /volume/CRYSTAL/ SHAPES instead of /volume/CRYSTAL. SHAPES and so on for the SAVE file and the SOUNDS subdirectory.

7. Write each block back to a copy of Crystal Quest

A tree directory of your new disk would look like this:

/volume	
CRYSTAL.SYS16	The game program
CRYS.PIC	The renamed title picture
/CRYSTAL	New subdirectory
SHAPES	CRYSTAL.SHAPES renamed
SAVE	CRYSTAL.SAVE renamed
/SOUNDS	CRYSTAL.SOUNDS renamed
SOUND01	The unchanged sound files
to	
SOUND25	

Now things are in a organized format. To move to a hard drive, simply copy CRYSTAL. SYS16, CRYS.PIC, and CRYSTAL. And copy the CRYSTAL.ICON to your ICONS subdirectory. To reset the High scores in Crystal Quest, do this:

1. Run/launch BASIC.SYSTEM
CALL-151
00<00/2000.00/2FFFZ
BSAVE CRYSTAL.SYS16, A$2000, B$48, L$1B7, T$B3

The scores are stored likes this (EX: 1548150 (27) (Big Mike)) at byte offset $48 in the file 50 81 54 01 27 "Big Mike" in ASCII with the high bit set. So zero'ing these bytes out resets the high scores. That's what the above BSAVE command does.

Softkey for...

Deja-Vu Shadowgate Uninvited

by Mindscape

Requirements:
768K Apple IIgs
3.5" disk copier
3.5" disk editor

The following edits will work for all three programs! The edits first kill the call to the copy

---

protection routine. Second, load the correct value for a pass condition and then BRanch Always to the continue code.

1. Make a copy the the game disk
2. Make the following edits to the game disk

Blk	Byte(s)	From	To
$D	$55	22 55 E7 00	AF 55 E7 00
	$59	CD BB BE	AD BB BE
	$5C	F0 08	80 08
$E	$153	22 55 E7 00	AF 55 E7 00
	$157	CD BB BE	AD BB BE
	$15A	F0 08	80 08
$23	$142	22 55 E7 00	AF 55 E7 00
	$146	CD BB BE	AD BB BE
	$149	F0 08	80 08
$24	$32	22 55 E7 00	AF 55 E7 00
	$36	CD BB BE	AD BB BE
	$39	F0 08	80 08
	$AD	22 55 E7 00	AF 55 E7 00
	$B1	CD BB BE	AD BB BE
	$B4	F0 08	80 08
	$189	22 55 E7 00	AF 55 E7 00
	$18D	CD BB BE	AD BB BE
	$180	F0 08	80 08
$27	$1EA	22 55 E7 00	AF 55 E7 00
	$1EE	90 70	EA 38
	$1F0	CD BB BE	AD BB BE
	$1F3	D0 6B	EA EA

3. Write the blocks back to the copy

Remember: If the copy protection routine will always fail on a copy, why run the code!

Softkey for...

Deja-Vu II

Mindscape/Icom

Requirements:
768K Apple IIgs
3.5" disk copier
3.5" disk editor

Deja-Vu II (DV2) is a continuation of the original Deja-Vu game with the same type of graphics interface. The same type of protection is used as on other Mindscape releases in this "series." However, the address and locations have changed for this release. Again these edits kill the call to the copy protection routine, load the right value and branch always to the continue code. To make a deproteced copy of Deja-Vu II follow these steps:

1. Make a copy of the game disk.
2. Make the following edits to the copy:

Blk	Byte(s)	From	To
$14	$188	22 00 00 02	AF 00 00 02
	$18C	CD 20 60	AD 20 60
	$18F	F0 08	80 08
$35	$8A	22 00 00 02	AF 00 00 02
	$8E	CD 20 60	AD 20 60
	$91	F0 08	80 08
	$185	22 00 00 02	AF 00 00 02
	$189	CD 20 60	AD 20 60
	$18C	F0 08	80 08
$36	$00	22 00 00 02	AF 00 00 02
	$04	CD 20 60	AD 20 60
	$07	F0 08	80 08
	$DD	22 00 00 02	AF 00 00 02
	$E1	CD 20 60	AD 20 60
	$E4	F0 08	80 08
$39	$16B	22 00 00 02	AF 00 00 02
	$16F	B0 70	EA 18
	$171	CD 20 60	AD 20 60
	$174	D0 6B	EA EA

3. Write the blocks back to the copy.

Play the game!

Softkey for...

FastBreak

Accolade

Requirements:
768K Apple IIgs
3.5" disk copier
3.5" disk editor

FastBreak (FB) is a new basketball game from Accolade. FB is like a new and improved GBA Basketball: Two on Two (Activision/ Gamestar). The major differences are: You now have three players per team, the men are about three times larger in FB, and you play on a half court at a time.

One thing both games have in common is the type of copy protection used. Like most games by Accolade (and it seems many companies) FB is using nibble counting on tracks $20 and $21. This is simple to find by searching for A2 20 A0 01 and it's found on block $85. The whole routine carries over into the next block and looks like this on the disk:

1DB:C2 20	REP #$20	Set 16-bit wide Accum.
1DD:AD 30 00	LDA 0030	Load 0030
1E0:48	PHA	Store on the stack
1E1:22 00 00 06	JSL 060000	Move CP routine to 00/ 8000
1E5:68	PLA	Get 0030 from stack
1E6:90 09	BCC 1F1 (+09)	If move was good, do nibble count
1E8:C2 30	REP #30	Otherwise set 16-bit Accum & index regs
1EA:A9 01 00	LDA #0001	Load a "fail" value
1ED:8D 26 00	STA 0026	Store in the COPY FLAG
1F0:6B	RTL	Return to caller
1F1:E2 30	SEP #30	Set 8-bit wide Accum & index regs
1F3:A2 20	LDX #20	Track $20

1F5:A0 01	LDY #01	and the side	
1F7:5A	PHY	Push side	
1F8:DA	PHX	Push track	
1F9:F4 C0 50	PEA 50C0	Push pntr to disk name	
1FC:F4 C0 50	PEA 50C0		
1FF:22 C1 00 06	JSL 0600C1	Count them up	
003:8D BB 50	STA 50BB	Temp store Accum (should = 00)	
006:68	PLA		
007:68	PLA		
008:68	PLA		
009:68	PLA		
00A:68	PLA		
00B:68	PLA	Pull all pushed info off stack	
00C:AD BB 50	LDA 50BB	Load the returned value	
00F:B0 4F	BCS 060 (+4F)	Carry set is I/O error or wrong disk	
011:8E BC 50	STX 50BC	Store low order of nibble count sum	
014:8C BD 50	STY 50BD	Store high order of nibble count sum	
017:A2 21	LDX #21	Now for track $21	
019:A0 01	LDY #01	same side	
01B:5A	PHY	Push side	
01C:DA	PHX	Push track	
01D:F4 C0 50	PEA 50C0		
020:F4 C0 50	PEA 50C0	Push pointer to disk name	
023:22 C1 00 06	JSL 0600C1	Count them up	
027:8D BB 50	STA 50BB	Temp store Accum (should = 00)	
02A:68	PLA		
02B:68	PLA		
02C:68	PLA		
02D:68	PLA		
02E:68	PLA		
02F:68	PLA	Pull all pushed info off stack	
030:AD BB 50	LDA 50BB	Load the returned value	
033:B0 2B	BCS 060 (+2B)	Carry set is I/O error or wrong disk	
035:8E BE 50	STX 50BE	Store low order of nibble count sum	
038:8C BF 50	STY 50BF	Store high order of nibble count sum	
03B:C2 30	REP #30	Set 16 bit wide Accum & index regs	
03D:AD BC 50	LDA 50BC	Load track $20 nibble count sum	
040:C9 6C 20	CMP #206C	Compare to low end bench mark	
043:90 1B	BCC 060 (+1B)	If too low then goto fail routine	
045:C9 03 21	CMP #2103	Compare to high end bench mark	
048:B0 16	BCS 060 (+16)	If too high then goto fail routine	
04A:AD BE 50	LDA 50BE	Load track $21 nibble count sum	
04D:C9 B0 1D	CMP #1DB0	Compare to low end bench mark	
050:90 0E	BCC 060 (+0E)	If too low then goto fail routine	
053:C9 79 1E	CMP #1E79	Compare to high end bench mark	
055:B0 09	BCS 060 (+09)	If too high then goto fail routine	
057:C2 30	REP #30	Make sure we're in full 16-bit mode	
059:A9 00 00	LDA #0000	Load a "pass" value	
05C:8D 26 00	STA 0026	Store in the COPY FLAG	
05F:6B	RTL	Return to caller	
060:22 A4 00 06	JSL 0600A4	Part of FAIL ROUTINE	
064:C2 30	REP #30	Make sure we're in full 16-bit mode	
066:A9 01 00	LDA #0001	Load a "fail" value	
069:8D 26 00	STA 0026	Store in the COPY FLAG	
06C:6B	RTL	Return to caller	
06D:00	00	50BB - Temp storage for Accum	
06E:00 00	0000	50BC - Track $20 nibble count sum storage	
070:00 00	0000	50BE - Track $21 nibble count sum storage	
072:46 41 53 54 42 52 45 41 4B		50C0 - FASTBREAK (disk name to check for)	

The routine is easy to follow and resembles most of the nibble counting routines that are used on the 3.5" disks. Tracing the program code back, I found a single call to the routine and made my patch there. The call is in the form of JSL025029, (22 29 50 02) then the flag location 0026 was loaded and checked for a value of zero. If 0026 was zero the game would continue, if 0026 was any other value the game would jump to a Pro-DOS quit routine.

One thing I noticed about this release of FastBreak (the one that was recalled). I bought the program and when I got home the FIRST thing I did was to make a copy and remove the copy protection. After I thought I had removed the routine (in fact I had) I booted the game and tried to play but could not figure out what was going on. So I booted the original to see if the same thing happened and the ORIGINAL would refuse to run! After reading the instruction manual I figured out how to start playing. The deprotected copy would run just fine while the original would quit before the game started. After some checking I found the copy protection would always fail on the ORIGINAL! However a kracked version was playable, this just goes to show what copy protection does for everyone, including the publisher.

Note: The version of FastBreak that you buy could have a slightly different copy protection, or be in a different place. To deprotect a different version, search for A2 20 A0 01 and back up about 20 bytes and list it. Compare the listed code to the above disassembly from $1E6 through $1F5 and find the flag, the LDA #0001 & STA 0026, where 0026 is the flag I'm referring to. Then, search for those two bytes, that is 26 00, if the flag was 0026. You should find code something like:

Block $A (on mine although it could be anywhere on yours)

```
1E0:22 29 50 02   JSL 025029   Do nibble count routine
1E4:AD 26 00      LDA 0026     Load the flag location
1E7:F0 04         BEQ 1EB (+04) Branch on zero (CP passed)
```

I found the above code about $3E0 bytes into the file FASTBREAK.SYS16. To remove the copy protection change the JSL (22) to LDA (AF), the LDA (AD) to STZ (9C), and finally change the BEQ (F0) to BRA (80). This skips the actual disk check, sets the flag to zero for an "original", then it always continues with the program. For the original non-working copies of Fastbreak:

1. Make a copy of the game disk

2. Make the following edit to the copy only:

Blk	Byte(s)	From	To
$0A	$1E0	22 29 50 02	AF 29 50 02
	$1E4	AD 26 00	9C 26 00
	$1E7	F0 04	80 04

3. Write the block back to the copy

Now you can actually copy the game to a subdirectory on a hard disk if you want. The only drawback to the game is you must hit reset to quit, as there is no built in quit that you initiate. This was kind of dumb as there is a ProDOS 16 quit call in the program that is used when a copy fails the copy protection routine. It would have been easy to check for a CTRL-Q and send the program to that quit routine, then you would have had a fully hard disk compatible game. Even though the manual specifically says you cannot upload FB to a hard disk on the GS, you can after you remove the copy protection. Once again the software publisher doesn't know what they are talking about.

Softkey for...

Final Assault
Epyx

Requirements:
512K Apple IIgs
3.5" Disk copier
3.5" Disk editor

Final Assault (FA) is a new mountain climbing game from Epyx. It uses nibble counting on tracks $20 and $21 for disk protection. The routine is much like those found on two other Epyx releases; Street Sports Soccer and California Games. However, there is one twist that helps hide the routine from the krackers. Searching for the standard A2 20 A0 01 will not work for this program. Epyx loads the registers in full 16 bit mode then switches to needed 8 bit wide registers to hide the routine. So for this release the A2 20 A0 01 has changed to A2 20 00 A0 01 00, when searching for this sequence you will find the following (block $354-$355):

1CB:0B	PHD		
1CC:3B	TSC		
1CD:38	SEC		
1CE:E9 F7 00	SBC #00F7		
1D1:5B	TCD		
1D2:69 EA 00	ADC #00EA		
1D5:1B	TCS		
1D6:AF 0E 00 02	LDA 02000E		
1DA:48	PHA		
1DB:22 00 00 0C	JSL 0C0000		
1DF:FA	PLX		
1E0:B0 56	BCS 038 (+56)	An error, goto fail section	
1E2:85 F0	STA F0		
1E4:A2 20 00	LDX #0020	Track $20 in 16 bit	
1E7:A0 01 00	LDY #0001	Side one	
1EA:E2 10	SEP #10	Switch to 8 bit wide registers	
1EC:5A	PHY	Push side	
1ED:DA	PHX	Push track	
1EE:A5 FF	LDA FF		
1F0:48	PHA		
1F1:A5 FD	LDA FD		
1F3:48	PHA		
1F4:22 C1 00 0C	JSL 0C00C1	Count them up	
1F8:85 F0	STA F0		
1FA:68	PLA		
1FB:68	PLA		
1FC:68	PLA	Pull extra values off stack	
1FD:86 EE	STX EE	Store first half of sum	
1FF:84 EF	STY EF	Store the second half	
001:C2 30	REP #30	16 bit wide accum & registers	
003:B0 33	BCS 038 (+33)	Had a disk error, then failed	
005:A2 21 00	LDX #0021	Track $21 in 16 bit	
008:A0 01 00	LDY #0001	Side one	
00B:E2 10	SEP #10	Switch to 8 bit wide registers	
00D:5A	PHY	Push side	
00E:DA	PHX	Push track	
00F:A5 FF	LDA FF		
011:48	PHA		
012:A5 FD	LDA FD		
014:48	PHA		
015:22 C1 00 0C	JSL 0C00C1	Count them up	
019:85 F0	STA F0		
01C:68	PLA		
01D:68	PLA	Pull extra values off stack	
01E:86 EC	STX EC	Store the first half of sum	
020:84 ED	STY ED	Store the second half	
022:C2 30	SEP #30	16 bit wide accum & registers	
024:B0 12	BCS 038 (+12)	Had a disk error, then failed	
026:A5 EE	LDA EE	Load track $20 sum in 16 bit	
028:8F 2B 0F 03	STA 030F2B	Store it	
02C:A5 EC	LDA EC	Load track $21 sum in 16 bit	
02E:8F 2D 0F 03	STA 030F2D	Store it	
032:22 A4 00 0C	JSL 0C00A4		
036:80 0A	BRA 042 (+0A)	Branch of the fail section	
038:A9 0C 00	LDA #000C	Load a failed value	
03B:8D 97 00	STA 0097	Store in the flag	
03E:22 A4 00 0C	JSL 0C00A4		
042:A8	TAY		
043:3B	TSC		
044:18	CLC		
045:69 0C 00	ADC #000C		
048:1B	TCS		
049:98	TYA		
04A:2B	PLD		
04B:6B	RTL	Return to the sender	

After checking the above code, I decided that the STA 0097 (8D 97 00) had to be some kind of flag. I then searched the disk for 97 00 for all references to 0097. This reveals the following code on block $447:

1A2:AD 97 00	LDA 0097	Load the flag	
1A5:D0 14	BNE 1BB (+14)	Anything but zero fails	
1A7:AF 2B 0F 03	LDA 030F2B	Load track $20 sum value	
1AB:C9 40 1F	CMP #1F40	Compare to benchmark (low)	
1AE:90 0B	CLC 1BB (+0B)	If too high, then fail	
1B0:AF 2B 0F 03	LDA 030F2D	Load track $21 sum value	
1B4:C9 D0 20	CMP #20D0	Compare to benchmark (high)	
1B7:F0 08	BEQ 1C1 (+08)	Equals is a pass	
1B9:90 06	BCC 1C1 (+06)	Or if it's higher, it passes	
1BB:A9 01 00	LDA #0001	Load a fail value	
1BE:8D 97 00	STA 0097	Store in the flag	
1C1:22 74,08 06	JSL 060874	Continue	
1C5:3B	TSC		
1C6:18	CLC		
1C7:69 16 00	ADC #0016		
1C9:1B	TCS		
1CB:2B	PLD		
1CC:6B	RTL	Return to the sender	

This section of code is the deciding factor in the copy protection routine. One, it checks 0097 for a value of zero. Two, it loads the results of the nibble count and compares them to the low end of the benchmark values.

If you think about it; 0097 should equal zero, then we need to get down to $1C1. I changed the LDA 0097 (8D 97 00) to STZ 0097 (9C 97 00) for the zero value, then the BNE 1BB (D0 14) becomes BRA 1C1 (80 1A) to get us down the continue section. Now, the one flag is set and the compare section is skipped over so the game will run.

The last addition I made was to bypass the actual disk reading routine, so I back traced the code and found a single call the first section of code I showed. This call is on block $3BC and is in the form of:

0CC:F4 D5 33	PEA 33D5	Push disk volume name	
0CF:F4 D5 33	PEA 33D5	D533 is relocated when loaded	
0D2:22 8B 07 01	JSL 01078B	Jump to the nibble count	
0D6:7A	PLY		
0D7:7A	PLY	Pull extra values off the stack	

With the main protections explained I would like to show you my steps:

1. Figure out what kind of copy protection is used. Epyx tends to use the 20/21 nibble counting scheme a lot. The only difference on the this release is the use of the 16 bit wide registers.

2. Find the copy protection routines. Okay, so we know we have to use A2 20 00 A0 01 00 to find the code. I tried to show what it is doing in the above disassembly.

3. Look for flags or special memory references. We find that 0097 is used as flag for the "horrible death" and insult routine. The program stores the nibble counting results at 030F2B and 030F2D.

4. Kill the calls to the copy protection. Changing the JSL 01078B to LDA 01078B will work for this step.

5. Force the flag checks to pass. Doing the STZ 0097 sets the flag, and the BRA 1C1 forces the code past the benchmark tests.

Now that you get some kind of idea of the needed steps for defeating the copy protection on Final Assault (and in general), you only need to make the patch permanent. To finish things up,

follow these easy steps:

1. Make a copy of the game disk.

2. Make the following edits to the copy only.

Blk	Byte(s)	From	To
$3BC	$D2	22 8B 07 01	AF 8B 07 01
$447	$1A2	AD 97 00	9C 97 00
	$1A5	D0 14	80 1A

3. Write the edits back to the copy.

Store the original in a safe place. Use the deprotected copy for playing.

The program is now kracked and the flag is captured! We get two more points, see how easy it is to rack up the points.

Softkey for...

Children's Writing & Publishing Center
The Learning Company

Requirements:
128K Apple II (with Double Hi-res)
3.5" disk copier
3.5" disk editor

The Children's Writing & Publishing Center (CWPC) is just what the title says, it's a desktop publishing program for kids. Because educational programs are used by children they should always be unprotected and easily backed up. Children (and adults) can have accidents that tend to render computer disks unusable! So I will show you how to make as many backups as you need.

The program comes on two 3.5" disks and is using a bad block on the program disk for protection. After a little code snooping, I was able to track down the call to the protection and disable it. To make a deprotected copy of CWPC follow these steps:

1. Make a copy of both disks (ignore any block errors on the program disk)

2. Using a block editor make the following edits to the program disk

Blk	Byte(s)	From	To
$3D8	$79	20 13 11	EA EA 18

3. Write the block back to the copy

Store the originals in a safe place. Give the newly deprotected copies to your kids to use!

Softkey for...

4th & Inches: Team Construction Disk
Accolade

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

4th & Inches Team Construction Disk (FITC) is a companion disk for the game 4th & Inches. With this new program you can change the players stats of the whole team, then save them to disk. You may also create entire new teams or add all the football teams you can think to FITC disk. Then you may select the teams you want to play with with. If you really liked 4th & Inches, but you where starting to loose interest, you should go out and get this program.

FITC does a standard nibble count on your original 4th & inches disk to see if you're using an original game disk. I'm not really sure if FITC runs the nibble count on it's self, but I know for sure it runs the check on the game disk. The protection routine is located on block $1BC of the disk and contained in the file START in the volume/SYSTEM subdirectory. The whole protection routine is as follows:

37:C2 30	REP #30	16 bit wide Accum.	
39:20 85 7A	JSR 7AB5		
3C:E2 30	SEP #30	8 bit wide Accum.	
3E:AD FD EC	LDA ECFD		
41:D0 2B	BNE 6E (+2B)		
43:EE FD EC	INC ECFD		
46:C2 30	REP #30	16 bit wide Accum.	
48:A2 01 02	LDX #0201	MMBootInit	
4B:22 00 00 E1	JSL E10000	Tool locator call	
4F:B0 6D	BCS C0 (+6D)	If error then bomb	
51:F4 00 00	PEA 0000		
54:A2 02 02	LDX #0202	MMStartUp	
57:22 00 00 E1	JSL E10000	Tool locator call	
5B:B0 61	BCS C0 (+61)	If error then bomb	
5D:68	PLA		
5E:8D FF EC	STA ECFF		
61:AD FF EC	LDA ECFF		
64:68	PLA		
65:22 62 E8 02	JSL 02E862		
69:68	PLA		
6A:B0 52	BCS BE (+52)		
6C:E2 30	SEP #30	8 bit wide Accum.	
6E:A2 20	LDX #20	Track $20	
70:A0 01	LDY #01	Side	
72:20 C1 EC	JSR ECC1	Jump to the nibble count	
75:8E 01 ED	STX ED01	Store half of result	
78:8C 02 ED	STY ED02	Store second half	
7B:A2 21	LDX #21	Track $21	
7D:A0 01	LDY #01	Side	
7F:20 C1 EC	JSR ECC1	Jump to the nibble count	
82:8E 03 ED	STX ED03	Store half of result	
85:8C 04 ED	STY ED04	Store second half	
88:AD 01 ED	LDA ED01	Load Track $20 result	
8B:C9 00	CMP #00	Check first half	
8D:AD 02 ED	LDA ED02		

90:E9 20	SBC	#20	Check second half
92:90 2A	BCC	BE (+2A)	If error then bomb
94:AD 03 ED	LDA	ED03	Load track $21 result
97:C9 00	CMP	#00	Check first half
99:AD 04 ED	LDA	ED04	
9C:E9 1F	SBC	#1F	Check second half
9E:B0 1E	BCS	BE (+1E)	If error then bomb
A0:82 2D 00	BRL	D0 (+002D)	Branch to continue code
A3:5A	PHY		Push side
A4:DA	PHX		Push track
A5:F4 02 00	PEA	0002	
A8:F4 F6 EC	PEA	ECF6	Push disk name/volume
AB:22 23 E9 02	JSL	02E923	Count them up
AF:8D FB EC	STA	ECFB	Store error flag
B2:68	PLA		
B3:68	PLA		
B4:68	PLA		
B5:68	PLA		
B6:68	PLA		
B7:68	PLA		Pull all extra values
B8:AD FB EC	LDA	ECFB	Load error flag
BB:B0 01	BCS	BE (+01)	If error then bomb
BD:60	RTS		Return to caller
BE:C2 30	REP	#30	16 bit wide Accum.
C0:A2 00 00	LDX	#0000	
C3:A9 DB DB	LDA	#DBDB	Load 2 STP's
C6:9D 00 00	STA	0000,X	Store them everywhere
C9:E8	INX		
CA:E8	INX		
CB:D0 F9	BNE	C6 (-07)	
CD:4C EB EC	JMP	ECEB	-= BOMB =-
D0:C2 30	SEP	#30	8 bit wide Accum.
D2:20 EB 79	JSR	79EB	
D5:E2 30	REP	#30	16 bit wide Accum.
D7:60	RTS		OKAY to run program!
D8:54 4F 4E 59	TONY		Disk name/volume

Well that's most of the protection routine, which is the exact same routines (different absolute addresses though) used on the 4th & Inches master disk. After checking this code I found the calls to it look like JSR EC55 (20 55 EB) on the disk. Searching for this sequence revealed two calls to the protection. After changing both JSR's (20) to LDA's (AD), I booted the disk and the program ran just fine. Applying the two disk edits resulted in a kracked working version of 4th & Inches Team Construction disk. The easy step by step method would be:

1. Make a copy of the game disk.

2. Make the following edits to the copy.

Blk	Byte(s)	From	To
$168	$80	20 55 EC	AD 55 EC
$190	$77	20 55 EC	AD 55 EC

3. Write the blocks back to disk.

Store the original in a safe place. Use the copy to work from.

Softkey for...

Geometry v1.0
Broderbund

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

Geometry is a great educational program from Broderbund/Sensei. They use the standard nibble counting routine on tracks 20/21 like they did for Showoff. To find the copy protection routine search for A2 20 A0 01, then list out the code that follows. Anyway, back tracing the routines twice, you end up with a single call to the routine. However, disk 2 would still ask you to insert disk 3 and if either disk/program was upload to a hard disk, they would ask for disk 3. I found the JSL right before the call to the nibble count checks to see if the disk online is called "/DISK3" so I killed that routine too! Now the programs no longer care what disk they are booted from or if it's an original.

1. Make of copies of disk 2 and 3.

2. Make the following edits:

On disk 2:

Blk	Byte(s)	From	To
$408	$8B	22 00 00 00	AF 00 00 00
	$93	22 E8 0D 00	AF E8 0D 00

On disk 3:

Blk	Byte(s)	From	To
$E7	$1C6	22 00 00 00	AF 00 00 00
	$1CE	22 E8 0D 00	AF E8 0D 00

3. Write the blocks back to the disk.

4. Optional: Upload the programs to your hard disk.

Softkey for...

Impossible Mission 2 GS
Epyx

Requirements:
512K Apple IIgs
3.5" Disk copier
3.5" Disk editor

Impossible Mission 2 (IM2) has been released in a Double Hi-res version for 128K apples and a full GS version. This new game is a futuristic continuation of the original Impossible Mission. The object of the game is to search through a multi-level building searching for parts

---

of puzzle. Of course you must avoid all the robots and jump from platform to platform, all of which makes for good game. To continue, I will explain the protection used on the GS version and I'll show how I went about defeating the routine. First, the game is released by Epyx, so I thought they might be using the old commercial nibble counting routine. Scanning for A2 20 A0 01 reveals the following on block $4A3 (on the disk):

35:8B	PHB		
36:4B	PHK		
37:AB	PLB		
38:AF 98 59 00	STA	005998	
3C:48	PHA		
3D:22 F5 69 00	JSL	0069F5	
41:FA	PLX		
42:B0 3A	BCS	7E (+3A)	
44:E2 30	SEP	#30	; 8-bit wide Accum & registers
46:A2 20	LDX	#20	; Load track number $20
48:A0 01	LDY	#01	; Side
4A:20 D7 69	JSR	69D7	; Go to the nibble count
4D:B0 02	BCS	51 (+02)	; Carry set means read errors
4F:80 04	BRA	55 (+04)	; Skip over the set fail
51:A2 FF	LDX	#FF	; Load a bad sum in X-reg
53:A0 FF	LDY	#FF	; Load a bad sum in Y-reg
55:C2 30	REP	#30	; Switch to 16-bit
57:8A	TXA		; Transfer hi-byte to Accum
58:8F D8 59 00	STA	0059D8	; Store in a storage area
5C:98	TYA		; Transfer lo-byte to Accum
5D:8F D9 59 00	STA	0059D9	; Store in a storage area
61:E2 30	SEP	#30	; Switch back to 8-bit
63:A2 21	LDX	#21	; Load track number $21
65:A0 01	LDY	#01	; Side
67:20 D7 69	JSR	69D7	; Go to the nibble count
6A:B0 02	BCS	6E (+02)	; Carry set means read error
6C:80 04	BRA	72 (+04)	; Skip over set fail
6E:A2 FF	LDX	#FF	; Load a bad sum in X-reg
70:A0 FF	LDY	#FF	; Load a bad sum in Y-reg
72:C2 30	REP	#30	; Back to 16-bit again
74:8A	TXA		; Transfer the hi-byte to Accum
75:8F DA 59 00	STA	0059DA	; Store in a storage area
79:98	TYA		; Transfer the lo-byte to Accum
7A:8F DB 59 00	STA	0059DB	; Store in a storage area
7E:22 99 6A 00	JSL	006A99	
82:AB	PLB		
83:6B	RTL		; Return to sender
84:5A	PHY		; Push side (this is 69D7)
85:DA	PHX		; Push track number
86:F4 00 00	PEA	0000	
89:F4 F0 69	PEA	69F0	; Push disk name location
8C:22 B6 6A 00	JSL	006AB6	; Count them up
90:8D F4 69	STA	69F4	; Store the Accum
93:68	PLA		
94:68	PLA		
95:68	PLA		
96:68	PLA		
97:68	PLA		
98:68	PLA		; Pull extra values off stack
99:AD F4 69	LDA	69F4	; Load the Accum again
9C:60	RTS		; Return to itself
9D:49 4D 32 00	IM2		; Disk volume/name
A1:00			; Storage area for 69F4

My first attempt (which works!) was to change the LDX #20 (A2 20) to BRA 51 (80 09) and the change the LDX #FF (A2 FF) to LDX #1F (A2 1F). This way a good value (at least passable) of $1FFF would be stored in the result location. Then I did a similar trick with the LDX #21 and the following LDX #FF. This works fine by never actually running the nibble count, but storing a passable value in the right locations thus "tricking" the routine into working. However, I decided to stick with my usual methods and do the following two things. First, find the call to the nibble count and disable it there. Second, find the compare routines and force them to continue.

The nibble counting routine is called once from block $3AE in the form of JSL 004AD8 (22 D8 4A 00). Changing the JSL (22) to a LDA (AF) will kill the call to the nibble counting routine. Now we must find the code that stores the read in values to known values. How do we find those routines? In the above disassembly the results are stored in 59D8 and 59DA. Simply search the disk for a reference to that location. Searching for D8 59 (59D8 in reverse, also I wasn't sure if they loaded it in long (LDA 0059D8 or AF D8 59 00) or short (LDA 59D8 or AD D8 59) mode showed the following section of code on block $3A6:

0FD:22 A5 18 00	JSL	0018A5	Get a bench mark value (1F50)
101:38	SEC		Set up for subtraction
102:ED D8 59	SBC	59D8	Subtract the read in value
105:70 03	BVS	10A (+03)	Is it too low?
107:49 00 80	EOR	#8000	Is it too high?
10A:10 08	BPL	114 (+08)	It's within the right range
10C:A9 01 00	LDA	#0001	Load a fail value
10F:8D 90 59	STA	5990	Store it in the flag

112:80 03	BRA	117 (+03)	Skip over the set pass
114:9C 90 59	STZ	5990	Zero = an original
117:22 F4 16 00	JSL	0016F4	Continue the game

So once again we find another flag waiting to be captured. If we don't get this flag, every time you try search for something, you're thrown back to title display.

Now this leaves us with the question of how to force the code to continue. Well we have several possibilities when looking at the above section of code. One, you could NOP everything from 105 to 10E and change the STA 5990 (8D 90 59) to STZ 5990 (9C 90 59); the only needed change is the STA to STZ. Two, change the LDA #0001 (A9 01 00) to LDA #0000 (A9 00 00). Three, change the BVS 10A (70 03) to BRA 114 (80 0D), which is what I choose to do.

Well to continue with the crack, do the following:

1. Make a copy of the game disk

2. Make the following edits to the copy only

Blk	Byte(s)	From	To
$3A6	$105	70 03	80 0D
$3AE	$3A	22 D8 4A 00	AF D8 4A 00

3. Write the blocks back to the disk

Store the original in a safe place. Use the copy to play from.

Softkey for...

Jack Nicklaus' 18 Holes of Major Championship Golf
Accolade

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

To remove the KEYWORD (Pirate) protection from Jack Nicklaus Golf follow these easy steps:

1. Make a copy of both of the Jack Nicklaus golf disks.

2. Make these edits to the COPY of the "game" disk only, never make any edits to the original!

Blk	Byte(s)	From	To
$92	$7F	F0 04	80 04
	$81	22 BA 03 00	AF BA 03 00

3. Write the block back to the copy

Store the original in a safe place with the manual. Optional: Upload the game to your hard disk. Play a game of Golf!

Softkey for...

Kinderama
Unicorn

Requirements:
1.25M Apple IIgs
3.5" disk copier
3.5" disk editor

Sure, right after I make a general comment like: "To krack all Unicorn programs search for A2 21 A0 01 20 and change it to 80 2C A0 01 20", Unicorn changed their protection to a bad block check. Anyway, I found three different kracks for the two disks. I forced the protection routine to pass on three different levels. Each edit on a different block will krack the program by itself (overkill!!) so take your pick or make them all.

1. Make a copy of the two disks.

2. Make the following edits on the copies only.

On disk 1:

Blk	Byte(s)	From	To
$220	$21	22 5B 33 00	AF 5B 33 00
	$25	B0 07	80 07
$224	$4A	22 CF 2F 00	AF CF 2F 00
$25C	$6C	22 48 38 00	AF 48 38 00
	$77	D0 03	80 0D

On disk 2:

Blk	Byte(s)	From	To
$4E2	$21	22 5B 33 00	AF 5B 33 00
	$25	B0 07	80 07
$4E6	$4A	22 CF 2F 00	AF CF 2F 00
$51E	$6C	22 48 38 00	AF 48 38 00
	$77	D0 03	80 0D

3. Write the blocks back to the copies

Softkey for...

Math Blaster Plus
Davidson

Requirements:
512K Apple IIgs
3.5" disk editor

Math Blaster Plus (MBP) has been released in a GS format and is a very nice educational program. The program is written in a version of Forth and is somewhat hard to follow. There is a routine to a "global" ProDOS 16 link (interface), that is, jump to the routine with the info and the routine modifies itself and makes the call. After much searching I did track it down, then with some help from Leh-Wen Yau I was able to come up with a nice little patch. The first edit kills that

---

call to the ProDOS 16 link at $26F. The second patch stores a zero (which means an error from the $26F routine) in the right place by changing the STA 0034,X to STZ 0034,X. A sneaky edit!

1. Make a copy of the Math Blaster Plus disk (ignore errors).

2. Make the following edits to the copy only.

Blk	Byte(s)	From	To
$9	$1BD	20 6F 02	AD 6F 02
	$1C2	9D 34 00	9E 34 00

3. Write the edits back to disk

Thanks again to Leh-Wen Yau. (Remember: Give credit where credit is due!).

Softkey for...

Qix
Taito

Requirements:
128K Apple II series
5.25" disk copier
5.25" sector editor

Warning: This article is written for those who like LONG articles

Qix from Taito is a micro adaption of the old arcade game. While the port over was not awesome, it was better than average (I'm still waiting for the GS version!). Anyway, the program is copy protected and the only way to back it up is to use EDD 4 plus (which I don't own) or krack it (that I can do).

The copy protection (CP) routine first searches for a CP header of D5 CC AA, then stores the following four bytes read from the disk. Then the routine proceeds to find the CP header again and compares the four bytes read to the four bytes stored. The trick is, because the bytes after the CP header have several zero bits in a row (making them invalid), the hardware CANNOT read them reliably. The whole CP routine is based on the fact that the invalid bytes will be read correctly less then 75% of the time! It loops through four times reading the same four invalid bytes after the CP header and increments one of four flags when a byte matches. Then another routine adds up the four flags and compares the value to $C (12 in decimal). If the four invalid bytes match more than 12 times then the program must be a copy. I will list the code used in the copy protection and comment it to making it easier to understand.

Now your bit copier reads the track once then writes it out to a blank disk. The problem is the bit copier will not reproduce all the extra zero bits so you end up with a non-working copy. Here is one of the control routines:

D19:AE DC 03	LDX	03DC	Get slot number IE: 6 for slot six
D1C:A0 01	LDY	#01	Get drive number
D1E:20 05 0A	JSR	0A05	Calculate slot/drive & store them
D21:20 11 0A	JSR	0A11	Go through the cp routines
D24:B0 05	BCS	0D2B (+05)	Couldn't find CP header or read error
D26:20 B9 0A	JSR	0AB9	Add flags & compare to $C
D29:90 1D	BCC	0D48 (+1D)	Less then $C matches, continue
D2B:4C C1 03	JMP	03C1	Wipe memory & reboot

Now here is the actual copy protection routines listed from $A00 to end:

A00-A04 is the data table for slot/drive & CP header.

A00:60			Slot number times 16
A01:01			Drive number
A02:D5 CC AA			Copy protection header

A05 is the entry point to calculate the slot & drive and store routine.

A05:8A	TXA		Transfer X to A
A06:0A	ASL		Multiply by 2
A07:0A	ASL		Multiply by 2
A08:0A	ASL		Multiply by 2
A09:0A	ASL		Multiply by 2 (times 16 overall)
A0A:8D 00 0A	STA	0A00	Store in slot number
A0D:8C 01 0A	STY	0A01	Store in drive number
A10:60	RTS		Return to caller

A11 is the disk read entry point and called from the game.

A11:A5 FE	LDA	FE	
A13:A6 FF	LDX	FF	
A15:85 FE	STA	FE	
A17:86 FF	STX	FF	
A19:AD 00 0A	LDA	0A00	Load Slot number * 16
A1C:8D B4 0B	STA	0BB4	Store it
A1F:4A	LSR		Divide by 2
A20:4A	LSR		Divide by 2
A21:4A	LSR		Divide by 2
A22:4A	LSR		Divide by 2 (from $60 to $06)
A23:09 C0	ORA	#C0	Make it into $Cx (or $C6 for slot 6)
A25:85 FF	STA	FF	Store it at FF
A27:AD 01 0A	LDA	0A01	Load drive number
A2A:8D B5 0B	STA	0BB5	Store it
A2D:A9 00	LDA	#00	Load zero
A2F:85 FE	STA	FE	Store at FE (FE=00 C6)
A31:A2 05	LDX	#05	We need to use six bytes
A33:BC B3 0A	LDY	0AB3,X	Load from "check slot" data table

Column 1 — Code listing (continued)

Addr	Opcode	Comment
A36:B1 FE	LDA (FE),Y	Load indirect+index, C600+Y
A38:CA	DEX	Decrement index
A39:DD B3 0A	CMP 0AB3,X	Compare to KNOWN value for 5.25" interface
A3C:D0 3A	BNE 0A78 (+3A)	If NOT EQUAL, then try different slot
A3E:CA	DEX	Decrement index
A3F:10 F2	BPL 0A33 (-0E)	Did we use all 6 bytes (X=$FF)
A41:A0 FF	LDY #FF	Load Y with FF manually
A43:B1 FE	LDA (FE),Y	Check C6FF
A45:F0 4B	BEQ 0A92 (+4B)	If it's zero then it's a 5.25" interface
A47:C9 FF	CMP #FF	Check to see if it's equal to $FF
A49:F0 2D	BEQ 0A78 (+2D)	If it is, then try another slot
A4B:69 03	ADC #03	Add 3 for "smartport" interface
A4D:85 FE	STA FE	Store value from CxFF+3 in FE
A4F:AD 01 0A	LDA 0A01	Load drive number
A52:8D AF 0A	STA 0AAF	Store it
A55:A2 21	LDX #21	
A57:90 07	BCC 0A60 (+07)	
A59:A9 01	LDA #01	
A5B:8D AF 0A	STA 0AAF	
A5E:D0 0D	BNE 0A6D (+0D)	
A60:AD AB 0B	LDA 0BAB	
A63:C9 01	CMP #01	
A65:D0 06	BNE 0A6D (+06)	
A67:AD AC 0B	LDA 0BAC	
A6A:F0 3E	BEQ 0AAA (+3E)	
A6C:38	SEC	
A6D:AD AF 0A	LDA 0AAF	
A70:8D B5 0B	STA 0BB5	
A73:CD 01 0A	CMP 0A01	
A76:D0 DD	BNE 0A55 (-23)	
A78:AD B4 0B	LDA 0BB4	Load slot number * 16
A7B:38	SEC	Get ready for subtraction
A7C:E9 10	SBC #10	Subtract $10 ($60 - $10 = $50)
A7E:B0 02	BCS 0A82 (+02)	CARRY SET unless $00 goes to $F0
A80:A9 70	LDA #70	Load $70 for slot 7

Subtract $10 from slot until it goes from $00 to $F0, then load $70 for slot 7. If you don't find a match ($70-$10 becomes $60) or the value stored at $A00, then with the CARRY SET, fall through to the return section of code. CARRY SET means had an error of some sort like: an I/O error (read error), couldn't find CP header, or couldn't find a 5.25" disk drive interface.

Addr	Opcode	Comment
A82:CD 00 0A	CMP 0A00	Compare to current slot
A85:D0 95	BNE 0A1C (-6B)	Not equal then go back and try again
A87:AD B6 0B	LDA 0BB6	This is how we get back from a JSR 0A11
A8A:85 FE	STA FE	B6, BB7, FE, FF not used in the actual program, but stored for the second call
A8C:AD B7 0B	LDA 0BB7	
A8F:85 FF	STA FF	to this routine.
A91:60	RTS	Return to caller, CARRY CLEAR = good read
A92:AD B5 0B	LDA 0BB5	Load drive number
A95:C9 03	CMP #03	Check to see if it's larger than 3
A97:90 05	BCC 0A9E (+05)	No, it's 1 or 2 then branch
A99:A9 01	LDA #01	Otherwise load 1 for drive one
A9B:8D B5 0B	STA 0BB5	and store it in drive number
A9E:20 C9 0A	JSR 0AC9	Do the actual CP routine
AA1:90 E4	BCC 0A87 (-1C)	Carry clear = found CP header, goto return
AA3:20 C9 0A	JSR 0AC9	Try again, might have missed it the 1st time
AA6:90 DF	BCC 0A87 (-21)	Carry clear = found CP header, goto return
AA8:B0 CE	BCS 0A78 (-32)	Couldn't find header, try another drive
AAA:38	SEC	Set the CARRY flag
AAB:B0 C0	BCS 0A6D (-40)	Branch on CARRY set (always taken)
AAD:60 03 00		Data, not sure for what though
AB0:C3 22 03		Data, again not sure for what
AB3:03 05 00 03 20 01		Data table for "check slot"

On most disk drive interfaces, where x=slot: Cx01=20, Cx03=00 and Cx05=03. CxFF is the address of the "dispatch" routine for ProDOS devices like 3.5" drives, SCSI drives, etc. Cx07 is the byte used to tell the difference between the types of drives (5.25" or 3.5") and SCSI. CxFB will tell if the drive can handle extended and/or SCSI calls. AB9 is the entry point for "add the flags & compare against $C" routine. This routine is called from game and not from within the CP routines.

Addr	Opcode	Comment
AB9:AD B0 0B	LDA 0BB0	Load read-match-flag-1
ABC:18	CLC	Set up for addition
ABD:6D B1 0B	ADC 0BB1	Add read-match-flag-2
AC0:6D B2 0B	ADC 0BB2	Add read-match-flag-3
AC3:6D B3 0B	ADC 0BB3	Add read-match-flag-4
AC6:C9 0C	CMP #0C	Compare to $C, CARRY CLEAR is less than
AC8:60	RTS	Return to caller

AC9 is the entry point the actually turns on the

Column 2 — Code listing (continued)

drive and starts to read for CP header and invalid bytes, called only within the CP routines.

Addr	Opcode	Comment
AC9:AE B4 0B	LDX 0BB4	Load slot number times 16
ACC:AD B5 0B	LDA 0BB5	Load drive number
ACF:18	CLC	Ready for an addition
AD0:6D B4 0B	ADC 0BB4	Add slot times 16
AD3:A8	TAY	Y=Slot+Drive
AD4:B9 89 C0	LDA C089,Y	Select the drive
AD7:BD 89 C0	LDA C089,X	Turn on the drive
ADA:A0 00	LDY #00	Load Y-reg with zero
ADC:8C A5 0B	STY 0BA5	Store in loop variable
ADF:20 73 0B	JSR 0B73	JSR to an RTS to let the drive come up
AE2:CE A5 0B	DEC 0BA5	to full speed. Loop through 255 times
AE5:D0 F8	BNE 0ADF (-08)	Did we finish the loop?
AE7:88	DEY	
AE8:F0 15	BEQ 0AFF (+15)	
AEA:BD 8C C0	LDA C08C,X	Load some bits from the drive
AED:DD 8C C0	CMP C08C,X	Compare to another read
AF0:F0 F5	BEQ 0AE7 (-0B)	If they match, the drive is not moving
AF2:A9 28	LDA #28	Load $28
AF4:8D A5 0B	STA 0BA5	Store it in the "find header loop", we should
AF7:8D A6 0B	STA 0BA6	find the CP header in $2828 disk bytes
AFA:20 74 0B	JSR 0B74	Routine to find the header
AFD:B0 03	BCS 0B02 (+03)	Carry set means we found the CP header
AFF:4C 6F 0B	JMP 0B6F	At B6F, set carry & turn off drive (failed)
B02:20 9F 0B	JSR 0B9F	Routine to read a byte off the disk
B05:8D A7 0B	STA 0BA7	Store it in "first-read-1"
B08:A9 00	LDA #00	Load zero to initialize "read-match-flag"
B0A:8D B0 0B	STA 0BB0	Store in read-match-flag-1
B0D:20 9F 0B	JSR 0B9F	Get a disk byte
B10:8D A8 0B	STA 0BA8	Store in first-read-2
B13:A9 00	LDA #00	
B15:8D B1 0B	STA 0BB1	Initialize read-match-flag-2
B18:20 9F 0B	JSR 0B9F	Get a disk byte
B1B:8D A9 0B	STA 0BA9	Store in first-read-3
B1E:A9 00	LDA #00	
B20:8D B2 0B	STA 0BB2	Initialize read-match-flag-3
B23:20 9F 0B	JSR 0B9F	Get a disk byte
B26:8D AA 0B	STA 0BAA	Store in first-read-4
B29:A9 00	LDA #00	
B2B:8D B3 0B	STA 0BB3	Initialize read-match-flag-4
B2E:A0 04	LDY #04	Set up to loop through four times
B30:A9 00	LDA #00	
B32:8D A5 0B	STA 0BA5	Set up to find the CP header somewhere in
B35:8D A6 0B	STA 0BA6	in the next $FFFF disk bytes
B38:20 74 0B	JSR 0B74	Find CP header
B3B:20 9F 0B	JSR 0B9F	Get a disk byte
B3E:CD A7 0B	CMP 0BA7	Compare to first-read-1
B41:D0 03	BNE 0B46 (+03)	Do the match?
B43:EE B0 0B	INC 0BB0	If yes, inc read-match-flag-1
B46:20 9F 0B	JSR 0B9F	Get a disk byte
B49:CD A8 0B	CMP 0BA8	Compare to first-read-2
B4C:D0 03	BNE 0B51 (+03)	Do they match?
B4E:EE B1 0B	INC 0BB1	If yes, inc read-match-flag-2
B51:20 9F 0B	JSR 0B9F	Get a disk byte
B54:CD A9 0B	CMP 0BA9	Compare to first-read-3
B57:D0 03	BNE 0B5C (+03)	Do they match?
B59:EE B2 0B	INC 0BB2	If yes, inc read-match-flag-3
B5C:20 9F 0B	JSR 0B9F	Get a disk byte
B5F:CD AA 0B	CMP 0BAA	Compare to first-read-4
B62:D0 03	BNE 0B67 (+03)	Do they match?
B64:EE B3 0B	INC 0BB3	If yes, inc read-match-flag-4
B67:88	DEY	Decrement the number of loops
B68:D0 C6	BNE 0B30 (-3A)	Continue if loop number not equal to zero
B6A:BD 88 C0	LDA C088,X	Turn the drive off
B6D:18	CLC	Clear carry = pass (found Cp header & bytes)
B6E:24		Used to make the following byte into BIT 38
B6F:38	SEC	Set carry = fail (didn't find CP header)
B70:BD 88 C0	LDA C088,X	Turn the drive off
B73:60	RTS	Return to caller

B74 is the entry point to read for the CP header, called within CP routines.

Addr	Opcode	Comment
B74:BD 8C C0	LDA C08C,X	Read for a byte
B77:10 FB	BPL 0B74 (-05)	Loop until the high bit is set
B79:CE A5 0B	DEC 0BA5	Dec disk bytes to read through (low order)
B7C:D0 05	BNE 0BA6 (+05)	Did we flip through the low order?
B7E:CE A6 0B	DEC 0BA6	Then dec the high order byte
B81:F0 1A	BEQ 0B9D (+1A)	If we flipped through high, then error out

Column 3 — Code listing (continued)

Addr	Opcode	Comment
B83:CD 02 0A	CMP 0A02	Compare to first CP header byte (D5)
B86:D0 EC	BNE 0B74 (-14)	
B88:BD 8C C0	LDA C08C,X	Get next byte
B8B:10 FB	BPL 0B88 (-05)	
B8D:CD 03 0A	CMP 0A03	Compare to second CP header byte (CC)
B90:D0 F1	BNE 0B83 (-0F)	
B92:BD 8C C0	LDA C08C,X	Get next byte
B95:10 FB	BPL 0B92 (-05)	
B97:CD 04 0A	CMP 0A04	Compare to third CP header byte (AA)
B9A:D0 E7	BNE 0B83 (-19)	
B9C:60	RTS	Return to caller
B9D:18	CLC	Carry clear means couldn't find CP header
B9E:60	RTS	Return to caller
B9F:BD 8C C0	LDA C08C,X	Load disk bits
BA2:10 FB	BPL 0B9F (-05)	Wait until the high bit is set
BA4:60	RTS	Return

BA5-BB7 is the data table for various things.

Addr	Data	Comment
BA5:FF FF		Number of disk bytes to read through
BA7:FF FF FF FF		First-read 1 through 4
BAB:FF FF FF FF FF		
BB0:FF FF FF FF		Read-match-flags 1 through 4
BB4:FF FF		First choice slot & drive
BB6:FF FF		Actual slot & drive

When looking at the control section, I decided to change the JSR 0A11 (20 11 0A) to CLC, BCC 0D48 (18 90 24) to always branch to the continue section without ever running the CP routines. I also found another call to 0A11 and calculated the correct branch on a pass and replaced it to.

Note: You could simply change the JSR 0A11 to NOP NOP CLC (EA EA 18), however then you MUST change the flag values (like I did) or kill the call to the flag check. The last thing I did was to make sure we don't find any hidden bombs later in the game. To do this, I found the four flags on the disk and changed them to values that would pass when added together. This is very important on other Taito release like Arkanoid GS, Arkanoid 2 GS, and Renegade IIe. After the second call to the CP routine is made and it passes, the routine is overwritten with game code.

1. Make a copy of the Qix game disk.
2. Make the following edits to the copy only:

Trk	Sct	Byte(s)	From	To
$00	$0D	$80	FF FF FF FF	01 01 01 01
	$0E	$38	20 11 0A	18 90 03
	$0F	$21	20 11 0A	18 90 24

3. Write the sectors back to the copy.

Store the original & play off the deprotected copy.

Advanced Playing Technique for...

Qix
Taito

The number of men are stored at $A5C, so for a never ending game change track $1E, sector $4, bytes $2F-31 from CE 5C 0A (DEC 0A5C) to EA EA EA.

While the game will not allow more then 5 men, you can start with 5 men by changing the A9 03 8D 5C 0A (TRK $0, byte $49) to A9 04 8D 5C 0A. The game counts your men from man zero to whatever. Then, when your man zero dies, the men flag goes to $FF and is checked to see if the high bit is set. When set (values of $80 to $FF), a BMI (Branch on MInus) instruction is taken and the game is over.

Softkey for...

Renegade
Taito

Requirements:
128K Apple II series
5.25" disk copier
5.25" sector editor

Renegade is nice conversion from the arcade game to Double Hi-res Apple II graphics. The game is copy protected just like Qix (also by Taito) with minor adjustments. The major difference in the copy protections are: First, if you do not set the flags (see the Qix article), the program messes up the shape table and the game will be unplayable. Second, Renegade is written under ProDOS 8. Third, and most important, Renegade use a KEYDISK system. In fact, the is the FIRST Apple IIe that I can think of that uses a keydisk.

Most of the time, when a copy protection routine fails, the program will just hang or reboot. With Renegade, you are allowed to insert the original in any drive! The best thing about this protection (other than it's about to be removed!) is the game is hard disk compatible.

1. Make a copy of both sides of the Renegade game disk
2. Make the following edits on side 1 of the copy

Trk	Sct	Byte(s)	From	To
$05	$08	$8A	0C 07 03 09	01 01 01 01 •
	$0B	$BD	20 62 28	18 90 2D

3. Write the sectors back to the copy
4. Rename LOAD.SYSTEM to RENEGADE.SYSTEM

Column 4

To upload to a hard drive (RAM drive, 3.5", etc), whatever, you can call it anything you like, but RENEGADE is a good choice. Now copy all the files except PRODOS into the new subdirectory from both sides of the deprotected game disk. You don't have to recopy a couple of the duplicate files on the back side, they are there so you don't have to flip the disk for every scene. Now you see why I renamed the system file, so you can easily tell which file to run to start Renegade. For an indepth study of the basis, theory, and ideas behind the copy protection used on Renegade, please read the Qix article.

Softkey for...

Silpheed
Sierra

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

Silpheed is a nice arcade shoot-em-up from Sierra. The only real problem I found was that before you can play a game you must identify an enemy ship. Which is, of course, Keyword (pirate) protection. To remove it make the following edits:

1. Make copies of both Silpheed game disks
2. Make the following edits to a copy of disk one only

Blk	Byte(s)	From	To
$1AA	$126	22 00 00 00	AF 00 00 00
	$12A	AD 00 00	EE 00 00
	$12D	D0 03	80 03

3. Write the edits back to the copy.
4. Optional: Upload the game to your hard drive!

Softkey for...

Sporting News Baseball
Epyx

Requirements:
128K Apple II series
5.25" disk copier
5.25" sector editor

Sporting News Baseball (SNB) from Epyx is a nice baseball game in double Hi-res on 128K Apple II series. Epyx is using the PROLOK copy protection scheme as on many other releases by Epyx and other companies. The routine has been well explained in previous issues of COMPUTIST, however, here is a quick run down.

First on track $00 there is a string of E7s. Each E7 has one or two added zero bits attached to it. When your disk controller card reads these bytes the added zero bits are ignored. Then upon bootup of SNB, a routine scans for the E7s, then in the middle of reading a byte, the routine clears the data latch and starts reading a new byte. Because the bits are no longer in sync with the E7 pattern, the added zeros becomes part of the new bytes read in. Then by comparing these "new" bytes to a list of expected read bytes, you can tell if the zero bits are present. By using this indirect method of checking, PROLOK (the scheme) can determine if the original disk is in the drive. Then to complicate things, the routine uses these bytes as keys in decrypting the RWTS that is used to load the game.

Now your bit copiers have a hard time telling the difference between a single or dual added zero bits. Most of the time, without special parameters, bit copy programs will not make an accurate back up. EDD 4 has a special parameter in the pre-analyze routine that searches for the E7s in memory and will add the zero bits when copy is being written. So to make a deprotected copy follow these steps:

1. Make a copy of both sides of The Sporting New Baseball

2. On a copy of side one make the following edits:

Trk	Sct	Byte(s)	From	To
$00	$01	$4E	A2 07 8A 95 F0	A9 FC 85 F0 85
			CA 10 FA A6 2B	F3 85 F6 A9 EE
				BD 89 C0 BD 8E
			85 F1 85 F2 85	C0 A9 0B 85 F0 F5 A9
			E7 85 F4	C6 F0 D0 0B EE 85 F7
			18 90 6E	

3. Write the sector back to the copy.

What the edit does is to store the right sequence of bytes in $F0 through $F7, then clears the carry and branches down to the section that decodes the rest of loader/RWTS. The patch is only twenty-five bytes long and except for maybe the exact same offset in the branch may be used on other programs that use this scheme. The code looks like this disassembled:

Opcode	Mnemonic	Comment
A9 FC	LDA #FC	Load FC
85 F0	STA F0	
85 F3	STA F3	
85 F6	STA F6	Store it in F0, F3 & F6
A9 EE	LDA #EE	Load EE
85 F1	STA F1	
85 F2	STA F2	
85 F5	STA F5	Store it in F1, F2, & F5
A9 E7	LDA #E7	Load E7
85 F4	STA F4	
85 F7	STA F7	Store it in F4 & F7

```
18      CLC         Clear the carry
90 6E   BCC +6E     Branch carry clear to
                    decode section
```

The one thing I found useful, but not actually listed anywhere, is the byte string that is read off an original. The string is:

FC EE EE FC E7 EE FC E7

This same string (stored in the right place) will work for many programs using this routine.

Softkey for...

Stickybear Talking Opposites

Optimum Resource

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

StickybearTalking Opposites (and Stickybear Talking Shapes) uses the same protection routine as Sticky Bear Talking ABCs outlined by myself in Computist in #66, page 6 and recently in Computist #68, page 17. Again there is a single call to the routine and killing it will result in a kracked copy.

1. Make a copy of the disk (ignore all read errors)
2. Make the following edit to the copy

Blk	Byte(s)	From	To
$EC	$15D	20 07 3A	AD 07 3A

3. Write the block back to the copy

Softkey for...

Superstar Ice Hockey

Mindscape

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

After getting the edits to krack this one from Stephen Lau, via a mutual friend, I added the edits on block $9 & $C. The other edits via Stephen allowed you to simply hit return to bypass the KEYWORD (pirate) protection. After the addition of my edits the requestor routine is also bypassed. Another victory for the Computist team... GO TEAM!

1. Make a copy of the game disk
2. Make the following edits

Blk	Byte(s)	From	To
$9	$5F	20 A3 1A	AD A3 1A
$B	$4	D0 13	EA EA
	$C	D0 0B	EA EA
$C	$11A	D5 00	D4 00
$38	$B7	D0 12	80 38
$A6	$B7	D0 12	80 36

3. Write the edits back to the copy.

Thanks again Stephen Lau for the krack and ground work!

Softkey for...

The Duel: Test Drive][

Accolade

Requirements:
1.25Meg Apple IIgs
3.5" Disk copier
3.5" Disk editor

Well it's another program by Accolade and they have been known to use the standard nibble counting routine, so I searched for A2 20 A0 01. This lead me to the actual copy protection routine, from there it was a simple matter of finding the calls to it. The calls to the nibble count are in the form of JSL 03C4E3 (22 E3 C4 03) and are in a couple of places. Making the following edits will remove the all copy protection from Test Drive][:

1. Make a copy of the game disk
2. Make the following edits to the copy:

Blk	Byte(s)	From	To
$169	$19B-19E	22 E3 C4 03	AF E3 C4 03
	$1F3-1F6	22 E3 C4 03	AF E3 C4 03
$191	$1D8-1DB	22 E3 C4 03	AF E3 C4 03
$199	$125-128	F0 16	80 16

3. Write the blocks back to the copy

Store the original in a safe place. Try not to get caught by the cops when you speed!

The edit on block $199 disables the routine that makes sure the master ID bytes match the gameplay disk's ID bytes.

Softkey for...

Where in the World is Carmen Sandiego GS v1.0

Broderbund

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

Where in the World is Carmen Sandiego has been released in a full GS format by Broderbund. And, of course, is copy protected. The routine is much like those found on Geometry v1.0 also by Broderbund. To find the copy protection search your disk for A2 20 A0 01. Back tracing the

---

routine twice results in a single call to the copy protection. Unlike Geometry, there is no routine that checks for a special disk name to be on line, so we only need to change a single byte.

1. Make a copy of disk two
2. Make the following edits to the copy of disk 2

Blk	Byte(s)	From	To
$69	$20	22 84 04 00	AF 84 04 00

3. Write the edits back to the copy

Softkey for...

World Geography v1.0

MECC

Requirements:
768K Apple IIgs
3.5" disk copier
3.5" disk editor

World Geography v1.0 by MECC uses a simple bad block check. After searching for 22 A8 00 E1 22 (ProDOS 16 block read) I found the disk check. I then traced it back to the caller (JSR 0B3B) and disabled it. I then checked the call right after it (JSR 0A3C) and check to make sure a 0027 (for I/O error) was returned after the read. So disabling both calls results in a deprotected version of World Geography.

1. Make copies of both World Geography disks
2. Make the following edits to disk one

Blk	Byte(s)	From	To
$68	$F5	20 3B 0B	AD 3B 0B
	$F8	90 05	80 03
	$FA	20 3C 0A	AD 3C 0A
	$FD	B0 23	EA 18

3. Write the block back to the copy

Softkey for...

Xenocide

Micro Revelations

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

Xenocide is a good GS game with nice sound and above average playability. However the game has a few minor and one MAJOR drawbacks. First, the game becomes very repetitive and there is no save game feature so you must play it until you die or complete it (it took me about 3 hours). After 3 hours of playing the game (with the cheats of course!), I found the winning title display a bit disappointing for the amount of work required to reach it. All these are minor problems compared to the outstanding, annoying problem (which prompts me to write this article) and that is its copy protection (CP).

Among other things, I was told that I couldn't remove the CP due to it's 50 checks and it is so tricky that you think you have the krack but later in the game it fails. This was as good a reason as any to krack the program! To steal a phase: "Let's get busy".

The protection is as follows: When the CP routine is called it will do the following: First, it scans for the smartport ID bytes and when found calculates the smartport dispatch address. It then stores this address in an extended STATUS and extended READ call routine. Then the program uses FWEntry (FirmWare Entry) tool call (tool call #2403) to make the calls. The CP reads all data in starting at 01/2000.

When the status call is made, it checks the DIB (Device Information Block) by comparing 01/2016 to 01 and 01/2017 to C0 to make sure you are using an Apple 3.5" disk (The CP, as written, is not compatible with the 3.5" Unidisk. This is ignorant programing as the Unidisk does support READ and extended READ commands. Unidisk support is yet another reason for deprotecting the program and "our" COMPUTIST magazine). The extended READ routine would read $C bytes into 01/2000 and then compare 01/200A to 08, this would only pass on the original. The CP routine is trying to read the Mac "tag bytes" for Block $4E1. Using Copy 2+, v8+ in the 3.5" sector editor, load in block $4E1 off an original and you'll see the eleventh byte of the "TAGS" row is 08. The CP is not very effective, as Copy2+, v8+ will make a working copy EVERY time using the standard sector copy. By the way: Due to this reason and the low level formats of 800k Mac and Apple 3.5" disks you can also use Copy 2+ on your GS to make copies of Mac disks.

If all the calls pass, 02/62DB would be set to 1E46 and this is checked later in the game (REMEMBER to capture the flag). Also there are checks to see if you placed a RTL at the beginning of the disk read and the "Insert Master...." routines.

The following edits will completely remove the CP and allow you to upload the game to your hard drive WITHOUT having to have the 3.5" disk in the drive at all times like the original requires you to. As an added bonus, the game now works just fine from a 3.5" Unidisk (I booted and played the game from a GS with a Unidisk). The reason Xenocide will now accept the use of a Unidisk is the program uses normal ProDOS calls (other than the CP) during the game.

Using cheats (provided at the end of this

---

article) I was able to play the whole game to the end three times and I found no problems what so ever, so this one is done.

Note: There are at least TWO versions of Xenocide released by the MR and most likely more. I'll show the edits for the two versions I have actually seen. The way to tell them apart are:

Version "A": 128 blocks, Dated 23-JUL-89 11:35 64996 bytes long.
Version "B": 129 blocks, Dated 13-APR-89 19:18 65085 bytes long.

Use the CATALOG command (or Copy][+ catalog with file length) and check under the created column.

Micro Revelations says this is a Key Disk copy protection, to which I say, Key Disk means you only need to have the original in the drive after the initial load and it's checked once. At that time, you may remove it and store it away until the next time you want to run the program. This copy protection is NOT like that, so it's a very annoying copy protection and NOT a key disk system. Finally, I was told (via third person) that Micro Revelations will send an unprotected version of Xenocide to registered owners on request.

1. Make a copy of the Xenocide game disk
2. Make the following edits to a COPY (by version):

Blk	Byte(s)	From	To
$7	$150	22 6F 09 00	AF 6F 09 00
	$154	90 0C	80 0A
	$156	22 6F 09 00	AF 6F 09 00
	$15A	90 06	80 04
	$15C	22 81 1D 00	AF 81 1D 00
	$160	80 EE	EA 18
$44	$E2	00 00	46 1E
	$E7	A9 00 00	A9 46 1E
$45	$B	22 6F 09 00	AF 6F 09 00
	$F	90 0C	80 0A
	$11	22 04 00 00	AF 04 00 00
	$15	90 06	80 04
	$17	22 81 1D 00	AF 81 1D 00
	$1B	80 EE	EA 18
$4B	15F	F0 01 60	EA EA EA
$62	$172	22 04 00 00	AF 04 00 00
	$176	90 0C	80 0A
	$178	22 04 00 00	AF 04 00 00
	$17C	90 06	80 04
	$17E	22 81 1D 00	AF 81 1D 00
	$182	80 EE	EA 18
$6A	$17C	D0 27	EA EA
$63D	$65	8F 06 00 00	AF 06 00 00

For version B:

Blk	Byte(s)	From	To
$8	$155	22 72 09 00	AF 72 09 00
	$159	90 0C	80 0A
	$15B	22 72 09 00	AF 72 09 00
	$15F	90 06	80 04
	$161	22 8B 1D 00	AF 8B 1D 00
	$165	80 EE	EA 18
$44	$FC	00 00	46 1E
	$101	A9 00 00	A9 46 1E
$45	$25	22 72 09 00	AF 72 09 00
	$29	90 0C	80 0A
	$2B	22 04 00 00	AF 04 00 00
	$2F	90 06	80 04
	$31	22 8B 1D 00	AF 8B 1D 00
	$35	80 EE	EA 18
$4B	$179	F0 01 60	EA EA EA
$5F	$18C	22 04 00 00	AF 04 00 00
	$190	90 0C	80 0A
	$192	22 04 00 00	AF 04 00 00
	$196	90 06	80 04
	$198	22 8B 1D 00	AF 8B 1D 00
	$19C	80 EE	EA 18
$67	$1B4	D0 27	EA EA
$634	$9D	8F 06 00 00	AF 06 00 00

3. Write the blocks back to the copy

Upload the game to your hard drive. Enjoy the absence of the copy protection!

That removes all the calls to the copy protection and sets the only flag (62DB; 0006 on the disk) to the correct value of 1E46. The two sections of code that scramble 62DB (the A9 00 00 8F 06 00 00), has been changed to either store the correct value or to load the value instead of storing it. The edits also force the check of the flag to pass no matter what value the flag has. Once again after the these edits are made you may upload the program to your hard drive and keep your original (and deprotected copies) in a safe place and never have to insert them in the 3.5" drive to play the game. So much for NOT BEING ABLE to krack Xenocide. Thanks to John M. Intondi for helping me with the block/byte locations for version B of Xenocide.

Advanced Playing Technique for...

Xenocide

Micro Revelations

Requirements:
a block editor with follow file function (Block Warden from ProSEL)

Apply ONLY to a backup copy. Use Block Warden and [F]ollow /XENOCIDE/ XENO.SYS16

Unlimited	REL byte	From	To	Notes
For version A:				
Men	$4DA	CE 17 00	AD 17 00	1
Fuel	$176A	CE 8F 15	AD 8F 15	1
Shields	$17A6	8D 95 15	AD 95 15	3
Missiles	$5298	8D AC 51	AD AC 51	2
Fireballs	$5266	8D AA 51	AD AA 51	2
Nuclear Mis.	$52EF	CE AE 51	AD AE 51	2

Lasers	$108C	CE 84 0E	AD 84 0E	3
	$1148	CE 84 0E	AD 84 0E	
Grenades	$1097	CE 82 0E	AD 82 0E	3
	$1111	CE 82 0E	AD 82 0E	
For version B:				
Men	$4DF	CE 1C 00	AD 1C 00	1
Fuel	$1774	CE 99 15	AD 99 15	1
Shields	$17B0	8D 9F 15	AD 9F 15	1
Missiles	$52A2	8D B6 51	AD B6 51	2
Fireballs	$5270	8D B4 51	AD B4 51	2
Nuclear Mis.	$52F0	CE B8 51	AD B8 51	2
Lasers	$1096	CE 8E 0E	AD 8E 0E	3
	$1152	CE 8E 0E	AD 8E 0E	
Grenades	$10A1	CE 8C 0E	AD 8C 0E	3
	$1160	CE 8C 0E	AD 8C 0E	

Note 1: Applies for the entire game
Note 2: Applies ONLY for the Hovercraft level
Note 3: Applies ONLY for the Cave & Bio-lab levels

Dave Goforth   WA

A BUG in Copy II Plus ver 9.0

"WARNING Volume NAME may be damaged. Please move data to another disk."

Have you seen a message similar to this lately? Have you recently purchased or used COPYIIPLUS ver 9.0? If you answered yes to both of these questions, then the first is the result of the second. COPYIIPLUS ver 9.0 has a bug in the sort routine. After sorting the main or ANY subdirectory, the program pointers are not updated and are no longer valid. This seems to only appear when using the finder, other program launchers (at least PRODOS direct commands, and Squirt) don't seem to notice this discrepancy. If you copy the file to another disk, the problem copies with it.

Hopefully you didn't use this on your hard drive as I did and completely backed up the hard disk, reformatted, and restored all the files to find that the problem still exists.

I called Central Point Software to let them know what I found. They returned my call the following day and said that they could not duplicate the problem. Since I knew there was a definite problem, I continued the investigation using other machines and other copies of ver 9.0 (to ensure that my copy was not the only one at fault). On ALL of the initial tests everything WORKED! My hardware at fault? No, I could make it fail every time no matter how I configured the system. Computer bad? No, everything else worked fine. What was I doing different than everyone else?

After two days of testing, retesting, and double checking both on my system and others, I could now make the disk work, bomb, and even unbootable, and just as easily repair it again on ANY system. The problem? CHEMISTRY! A combination of using the sort routine in Copy II Plus, Bird's Better Bye, and GS/OS will result in disaster (especially for a hard disk). I have my system configured as follows:

1. Patched ProDOS with Bird's Better Bye. This quickly launches my backup routine and exits to a selector menu. This way I can quickly boot up, run any ProDOS 8 program and exit - no waiting for GS/OS's finder to load (especially if I don't need it).

2. I've renamed ProDOS from the GS/OS system disk to GSOS. If I do want to use GS/OS's finder, I select GSOS from the menu and (wait) now the desktop is displayed and I can use ProDOS 16 files.

The Beginning of Tragedy

GS/OS, NORMALLY does not care about parent pointers, but if launched from Bird's Better Bye menu selector, it now cares a great deal and all it needs to detonate is for ANY folder to be moved within ANY directory (or subdirectory) by Copy II Plus's sort routine and the message "Volume VOLUME.NAME may be damaged. Please move data to another volume. <OK>" will appear. If the SYSTEM folder (or any of its subdirectories) have been relocated, the disk will display this message three times then "Sorry, system error $0051 occurred while loading the ??????? files. <RESTART>" and the disk will be unbootable if GS/OS is launched with Bird's Better Bye.

Fixing the Disk

There are three known ways to fix the disk back to perfect condition.

1. Rename the patched ProDOS (with Bird's Better Bye) to ProDOS.Bye (or anything you like) and rename GSOS to ProDOS. Reboot this disk (it MUST contain the GS/OS operating system with all system folders and icons), open the disk to display it's contents (OPEN from the file commands or click the disk twice), then close the disk's window (CLOSE from the file commands or click on the upper left box of the disk's window), and drag the disk to the trashcan to eject it. ALL parent pointers are now correct (even those buried in subdirectories). NOTE: This method will ONLY work on the boot disk. If you insert another disk, open, close, and eject it, that disk will NOT be fixed, even if it's a system disk. It MUST be the booted disk. Since the disk within a hard drive cannot be ejected, this will not work on a hard disk.

2. The EASIEST and FASTEST way to correct (fix) any disk (including hard drives) is to use Procel's Mr. Fixit, Quality Software's Bag of Trix 2, or any other disk utility that can check and correct parent pointers. Both of the above mentioned programs have been tested and work fine.

3. The third, and probably the most difficult, is to use Copy II Plus's sort routine and put the directories back in their original sequence. Did you write down or print out the catalog before reorganizing or do you have a photographic memory, no? If you can't use step 2, hopefully the disk is bootable and you can use step 1.

Final notes: This problem will usually never appear for most people, since it requires that GS/OS be launched after Bird's Better Bye has been run. If you REBOOT GS/OS or use any other program launcher/selector exclusively, then you'll probably never know that Copy II Plus's ver 9.0 sort routine has not updated your directory parent pointers and everything will work fine (unless your application specifically looks at 'hem; and most don't).

To Rich Linville: Use AMDOS if you need DOS 3.3 on a 3.5" disk.

To Ann Onymous: I can transfer IBM files to Apple DOS, so go ahead and send your material to COMPUTIST on IBM disks.

The Dark Phoenix   UT

Well, to put it plain and simple, after receiving issue #66 I was shocked! I would never have guessed that a magazine as good as COMPUTIST only had about 4000 subscribers, so that means that about 3,000 are faithful renewers, that's puzzling.

Now that I'm done kissing up, I can tell you what I propose. I like to think of it as taking the ice cream and cake in one bite. As much as a hate to say this, I believe that the first step is going to the tabloid format, I personally HATE the tabloid format, but its cost-effective. Second, I would go to a two-week publication rate (gradually) and have subscription renewals at 6-12 months. Then I would raise the price $1. That's not so bad and you would probably only lose about 50 subscribers. Then I would see what I could do about getting in book stores and computer stores (hardware of course) and supermarkets. I never even heard of such a magazine until a friend told me about it! Which goes to show you how hard it is to find COMPUTIST. Fourth, (IF NEEDED!) do away with the mac section all together and make more room for editors comments, reviews, etc. Look at InCider, that's all it is and it has a HUGE subscriber list.

If you can avoid it at all, please don't keep going on the tabloid format, its so annoying, not to mention unprofessional looking.

I liked your idea of giving brownie points to all people who bring new subscribers, but I also think that it would be a good idea to have a raffle every three months for a piece of hardware (modems, hard drives, memory cards, enhancements, etc.) and every new subscriber you bring in you get one entry ticked into the raffle, the result = the more new subscribers you get, the better chance you have of winning! or you could do a 1$ an entry raffle, non profit of course, and make it open ONLY to subscribers, that would be a catch that sure to bring in 400-500 new ones each year.

The fact that COMPUTIST remains unadvertised is a fault that is loosing COMPUTIST thousands of subscribers. If COMPUTIST were to just send fliers around, to the many who have accidentally destroyed originals, they would have many new subscriber.

I also believe that if COMPUTIST got more in-depth on HOW to find a sector error and correct it, there would be many new subscribers, because beginners are interested in HOW not WHAT to do! (take it from a professional beginner). I believe that a contest, to see who can explain this would be the best way to pursue this! Have a most wanted (speaking of which, if anyone knows the softkey to Deathlord, I need it badly, my original is getting trashed) softkey, and a contest on who can de-protect it, not only the fastest, but the most in-depth. have the beginner readers write in and cast their votes.

Carl D. Purdy   MT

First let me say that I enjoy and look forward to each and every issue of Computist each month or every six weeks now. Enclosed please find my subscription to the Computist Club. I think that is a GREAT idea. I know that I have questions that could be answered on a board much faster than in the mag at times. Keep up the good work.

I would like you to place some things on the Most Wanted List. They are Where in the World is Carmen Sandiego. I know that there is a softkey in an earlier issue but it doesn't work on all of my computers. The keyed program will boot and run on my oldest IIe (with mother board A) but will not boot and run on my newer IIe (mother board B) nor will it run on a Laser 128. Could someone out there help me. Another program I would like to unlock or otherwise back up is Keyboarding Klass by Mastery Development. It is on a PRODOS like disk called PAJDOS. I can get a catalogable disk (from PRODOS) but none of the files will run. Finally I would like a softkey for Gertrude's Puzzles.

---

⑦ If any one out there knows why that a program backed up with Wild Card will not run on a Laser 128 please help!

Ron Powers   WI

⑦ For Edward Teach (or others): I am having trouble cracking 2 programs. One is Design Your Own Home:Landscape, by Abracadata. I have tried all procedures listed for this one in past issues, but none will even copy my version of the disk. The other program is Success With Reading, by Scholastic.

⑦ My other question regards a graphics program called Super Print!, by Scholastic. Thanks to Computist, I have successfully cracked this one. The program allows the printing of graphics in several sizes. However, the 5 ft. tall graphics are separated into a file called "Posters." The problem is that there is a very limited supply of these large-sized graphics with the program. Is there anyone out there who knows how to convert the regular-sized graphics into poster (super) sized?

Here are three softkeys that I was able to perform with the help of past articles dealing with these same publishers:

Softkey for...

Boppie's Great Word Chase
DLM

Requirements:
A way to reset into the monitor
Super IOB 1.5 with Newswap Controller

1. Capture the program's RWTS

2. Install the Newswap Controller and run Super IOB

3. A fast DOS may be installed

Softkey for...

Surveys Unlimited
Learning Well

1. Poke out end checks and use COPYA to copy the disk.
POKE 47426,24
RUN COPYA

2. Transfer files to a DOS 3.3 or Fast DOS disk.

Softkey for...

Campaign Math
Mindscape

I followed Jack Moravetz's softkey for Ace Reporter, on p. 19 of COMPUTIST #64:

1. Copy with any copier that ignores errors.

2. Make the following sector edit:

Trk	Sct	Byte(s)	From		To
$01	$01	$00	A2		60

Keith   WA

Softkey for...

Superprint v1.2
Scholastic

Requirements:
Apple IIgs with 2.x ROM upgrade
Two ProDOS formatted disks
BASIC.SYSTEM from a ProDOS master disk
Copy II Plus or any ProDOS file copier

Although Superprint has been cracked in several previous articles, the version I have was not readable by COPYA. When I booted the original disk, the ProDOS credits were displayed, so I decided to catalog the disk and see what the files were. However, when I tried to catalog with Copy II Plus, the program gave me "error reading track 11," an error that one usually gets from DOS disks, not ProDOS disks. This would mean that the program when it first boots (track 0) fools Copy II plus into thinking that it runs under DOS rather than ProDOS. Another way to say this is that Scholastic has used a short DOS-like program to boot the disk, then loads ProDOS and the catalog directory. Since they are not in the normal places on the disk, Copy II Plus doesn't see them.

The one other clue I had was that I made a backup using the autocopy feature and Superprint parameters on Copy II Plus v8.3. Although this made a working copy, I got a few that did not work. The program would boot, then reboot almost immediately, and continued to do this endlessly. All this occured before the ProDOS screen appeared. This meant that a signature check was being done, and that it appeared before the ProDOS was loaded.

With all these assumptions, I set out to crack the protection. I decided to try the system used by Dan Halfwit for Pirates and Pirates GS that appeared in Computist 61 (an excellent pair of articles). After setting "visit monitor", I booted the original Superprint disk, and quickly went to monitor. This allowed their DOS startup to load and run the signature check. I then scanned for the Jump to $2000 that had been loaded, and replaced it with a jump to monitor. This gives you a stopping point in the program before it begins to run ProDOS, since the beginning point of ProDOS is always at $2000. I then left the moni-

---

tor, let the rest of ProDOS load, and then the program jumped back into the monitor because of the stop I had placed above. I then loaded BASIC.SYSTEM as described below (BASIC.SYSTEM will be renamed SP.SYSTEM, but for this discussion I will continue to call it BASIC.SYSTEM). With ProDOS and BASIC.SYSTEM in place, you should now be able to catalog the original Superprint disk. The catalog directory was loaded with ProDOS, and when you are all done, if you look at the Superprint files, you will see one called "disk.info". This contains the directory information.

To copy the files, I next loaded Copy II Plus using the ProDOS smart command. This is done using -UTIL.SYSTEM. This gives you the familiar Copy II Plus menu and you use the "copy files" to copy ALL the files, including ProDOS. You can copy from drive 1 to 2, from 5.25 to 3.5, or from 5.25 to RAM (slot 5, drive 2 on Copy II plus), then later from RAM to a 5.25 or 3.5 disk. Using the RAM is useful if you only have a single 5.25 disk drive, but you have to know how to go into the control panel and set your RAM. (To do this, go to the control panel OA-CNTL-ESC, select RAM DISK, and set the maximum ram in 832K, and leave the minimum to OK.)

I did not need to do any block edits like Dan Halfwit did, because the protection on this disk is part of the DOS load program, and the above system doesn't copy the jump to the protection system.

Cookbook

1. Format in ProDOS both your blank disks using CopyII Plus, and use the volume name SUPER.PRINT. Put one copy aside. With the other one, use CopyII Plus File Copy to copy BASIC.SYSTEM to the disk. When this is completed, then go to the RENAME feature on CopyII Plus, and rename the volume to SP.SYSTEM. Label this disk to distinguish it from the one without SP.SYSTEM. You can find a copy of BASIC.SYSTEM on your ProDOS users disk, and on a lot of software programs. I have made a special disk which has the current version of ProDOS and of BASIC.SYSTEM on it, so that they are easily available to file copy onto my disks.

2. Now set up your IIGS. To get into the monitor, boot up the computer (you don't need a disk in place), then hit OA-CNTL-ESC once or twice fairly soon after the startup, and you will see the Basic prompt. If you miss, try again since this always works. Then type CALL -151, which takes you to the monitor, type a # sign, and hit RETURN. This installs the visit monitor CDA. Now put your Superprint original into the disk drive (be safe and have it write-protected).

Type in C600G at the monitor prompt to boot the disk. Very shortly after the disk starts (one or two clicks), press OA-CNTL-ESC and this will take you to the control panel. This step is one place where errors can occur. If you do it too soon, then the DOS boot and signature check don't get done. If you do it too late, too much ProDOS gets loaded and the program runs. This is not a fatal error, it just means that you may have to try this step several times. Once the ProDOS screen appears on your computer, you know that you are too late. You can tell if you are too early in a subsequent step that I'll mention. If you have to retry, just reboot with OA-CTRL-RESET, rather than go back to the monitor and typing C600G.

3. You are now at a menu. Select Visit Monitor, and hit return. You will see the monitor prompt. You now need to scan for the bytes which tell the program to jump to ProDOS (JMP 2000, which is 4C 00 20). Enter \4C 00 20<800.1FFFP and hit RETURN. On the screen you will see 00/08FC:. This is the location that the string starts at. If you type 08FCL then hit RETURN, you will get a listing of the bytes. At 08FC you should see 4C 00 20. (Note that the reason you did the scan is to find the location. You might have something different than 08FC, and should use whatever your scan gives you). Replace the 4C 00 20 with 4C 69 FF which will cause it to jump to the monitor rather than ProDOS. To make the replacement, type 08FC:4C 69 FF, then hit RETURN. Now hit CNTL-Y and RETURN. The program will go a little further, then a monitor sign will appear. What it is doing is finishing the loading of ProDOS and the directory file, then it wants to jump to ProDOS to run the program, but instead jumps to the monitor because of the bytes we have changed.

4. You now have the ProDOS and directory loaded. To be able to catalog and load CopyII Plus, you need to load the BASIC.SYSTEM. When ProDOS runs, it will load the first file whose name has .S SYSTEM. At this point take out your Superprint original, and insert the ProDOS disk you made that has BASIC.SYSTEM on it(you renamed it to SP.SYSTEM). Type 2000G which will start the program running at ProDOS. This will load the BASIC.SYSTEM, and shortly thereafter you will see the basic prompt].

5. At this point you get to see if you did everything right. Take the disk out and insert the Superprint original. Type CATALOG, hit RETURN and you should see a list of files scroll by. If this works, then smile. If not, then you need to go back to step 2, where you boot the Superprint disk (you do not need to do the # again).

6. Now you load CopyII Plus. Put your Cop-

---

yII Plus disk in the drive, type -UTIL.SYSTEM, hit RETURN. This loads the program, and you will see the familiar CopyII Plus menu. Take out the disk, and insert your Superprint original. Select COPY, then FILES. The source is Slot 6, Drive 1. The target is Slot 6, Drive 1 or Slot 5, Drive 2, if you only have one 5.25 drive (I prefer that latter, which copies to the RAM as discussed above). It is easier for the beginner to copy to Slot 6, Drive 2 if they have two 5.25 drives, or use Slot 5, Drive 1 to copy it to a 3.5 drive (you can always copy from the 3.5 to 5.25 later if you want). At this point all the Superprint files should be visible. Select all the files. You will see the first file highlighted. Hit the RETURN key and the number 1 appears next to it. Hit the arrow key to go down to the next file, hit RETURN and the number 2 appears next to the second file. Repeat this until all the files are selected. At this point the Superprint original should be in the source drive. The second ProDOS disk that you formatted with and named SUPER.PRINT should be in the target drive (this disk is the one that does not have BASIC.SYSTEM on it). Type G, and the copying should begin. Go have a cup of coffee, since this part takes a few minutes.

7. When the copying is done, you should now have a working copy. The protection is removed. CopyII Plus can catalog it, and make copies using the regular disk fast copy procedure. Put your original in a safe place.

8. The back side of the original, and both sides of the other disks can be copied using CopyII Plus and the disk copy program. They are not protected.

⑦ Issue 65 contained several excellent articles on converting the ProDOS word processor files to DOS. How about putting these articles on a disk for new subscribers, so they will find it easier to make contributions.

It really isn't necessary to convert Pro-DOS word processor files to a DOS format. Just send what you have and write on the disk what program you used. We'll do the conversion here. ....RDEXed

⑦ I am impressed when articles contain long printouts of code that are annotated. How can I capture code the same way for my articles? I'd hate to have to type it all manually.

⑦ To Ed Teach, Brian Troha, or one of you other great hackers; How about an article on COPYA and what types of protection are used to keep it from reading a disk. Many of the new disks cause the "unable to read" error to appear when using COPYA, even when DOS is altered to ignore errors, including prologs, etc. What is done to make the disk unreadable, and what changes can be made to make it readable.

Enclosed is my membership for the Computist Club. I like the idea for giving people a chance to win free software. The thing I don't like is that I will have to wait longer between each issue. For those of us who live by a sense of order, perhaps you can publish a list of anticipated mailing dates so that we can know generally when to expect the newsletter (give or take a few weeks <grin>).

Also for those of us who send in a disk, getting back a Computist disk is not always that attractive. Perhaps an alternative would be too include Public domain software or Shareware. You could list disks in the magazine or just surprise us (I think a choice would be nicer). Things which could be included are the monthly disk from Applelink, or things such as fonts, games, and graphics. For members who do not have a modem, they could find this attractive, and you could get the software without much expense. If you need help finding stuff, let me know.

I also have a question regarding the proposed Computist bulletin board. The idea is a good one, but it should take the costs of telephone connections into consideration. Would it be less costly to work through Genie or Applelink or Compuserve? If these phone costs are too high, then the members will not use the bulletin board very much.

In regards to your advertising, I had seen it several times before finally taking a fling. I thought the newsletter was a fly-by-night thing that either would not last, or would be too hard too use. Your ads need to emphasize ease of use, how long you have existed (I was surprised when I joined), and perhaps list the names of some recent deprotects that have been published of "big hits." You might even include a free single subscription to people to let them see how good the newsletter is.

Softkey for...

Magic Spells
Learning Company

I was having a devil of a time as I had tried all of the softkeys in Computist without success. It turns out that the softkey is really quite simple. Back up the disk with Advanced COPYA, then get into PRODOS. Load the Startup program. Eliminate the two CALLs from the ends of lines 45 and 61. That is all there is to it. This is another bit of help that I received from a friend, Otis Thompson down in Dillon, Montana.

Stephen M. Caraco   MA

⑦ I need help copying Arkanoid by Taito. The very first thing I do when I get a new game

(before I even THINK of playing it) is to catalog the disk to get an idea of what types of files are on it, the operating system, how much of the disk is full, etc. The next thing I do is to just make a straight copy of the disk, and boot it up, waiting very impatiently for the disk to crash. If it boots fine, I know I've saved hours of work. If it doesn't, I try to figure out the method of protection. Well, I've come across a tough protection scheme—this one is by Taito Software. The disk is Arkanoid.

When I first booted the copy, it seemed to boot fine. Then, it asked me to insert a key disk and press any key to continue. Although I know in my subconscious that it would never work, I tried pressing return a few times hoping that the program will give up asking after a few times. What did I have to lose? My patience. This is where it all began.

The next thing I did was to break out my Copy II Plus, version 8.4. I was shocked to very easily find the text "PLEASE INSERT THE ARKANOID KEY DISK INTO A DRIVE AND PRESS A KEY" using the sector editor and searching for text. This text can be found spread out over blocks $024A and $024B. With other games, such as BUBBLE GHOST (whose key text is "WHAT LEVEL IS THIS?"), searching for the key text is impossible. So this is where I continued to work on the disk.

Since I am not as familiar with copy protection as some people seem to be, I am now lost. I did try NOPing out some of the jumps on that track, and changing a few 80's to F0's at random, but nothing seems to work. I've changed various things, one at a time, and nothing works. Just before I gave up, I found the text NOTICE: (which appears in the box that asks you to insert the key disk) on block $0249. I found a jump (20 1C 11) immediately before this, and NOPed it out, and all this seemed to do was keep the text from appearing in the box. However, the program still crashed, beeped, and I couldn't play the game!

I do know enough about copy protection to know that there is only a simple byte change that will disable the key disk check on the ARKANOID game. I just can't seem to find it! Perhaps I am looking in the wrong place....

With this, I looked through each block of the entire disk, one at a time. As I approached blocks $0291-$0292, I came across the text ...BUT ONLY TO BE TRAPPED IN SPACE, WARPED BY SOMEONE...SOMEONE AS DEVIOUS AND WARPED AS A SOFTWARE PIRATE. IT'S GOING TO BE A LONG NIGHT DUDE. A REAL PIRATE WOULD ADD HIS OWN MESSAGES INSTEAD OF VANDALIZING THESE DIMENSION-CONTROLLING FORT. This doesn't make much sense, but perhaps the protection is located somewhere on these two blocks.

I've just about given up. I hope this information will prompt someone with a copy of this great game to figure out the softkey for it. Oh, by the way...I was able to easily get into the control panel and use the VISIT MONITOR desk accessory immediately after being asked to insert the key disk. The only problem is, I have absolutely no idea what any of the stuff means when I use the VISIT MONITOR accessory.

Somebody, please help. I would give anything to be able to copy this game.

Softkey for...
California Games GS
Epyx

The softkey in COMPUTIST #64 (page 37) did not work for me. But this simple, one-byte change made my California Games copyable both onto a hard disk, and bootable from a RAMdisk! Here's the simple change:

Blk	Byte(s)	From	To
$0D2	$095	??	80

⑦ Now I have a question for all you hackers out there: How in the H*LL do you figure out which JSR and JMP's to change when cracking a disk? It seems a little too simple the way you describe it in your softkeys. I'd like to know how you know which ones have to do with the protection, and how you know what to change them to! If someone out there is reading this, please publish a checklist for IIgs software which could give me a list of HEX strings to search for on a typical GS game, what they mean, how to find them, and how to change them. I do know a FEW already, but they don't seem to help. For example, I know that on 5.25" disks 4C 00 C6 means to REBOOT. I'd like to know some standard strings on GS software, and what they mean.

I definitely believe that deprotecting a disk (or at least trying to) is much more fun than playing the actual game! Well, have fun. I'm sure I'll think of a thousand more questions as soon as I put this in the mailbox.

John Ianzano   NY

⑦ I am a new Computist subscriber and have the following problem; I'm probably the last person in the United States who's still using a program called Screenwriter 2.2 and I want to save my text files to my 3 1/2 inch disk drive. I've used the softkey for Screenwriter which was published (issue #9 & 10) and now have a

COPYA-able version. The problem is that I still can't save text files to my 3 1/2 inch drive even when I use a modified Dos 3.3 to format my diskette (IE.; Profix, UniDOS). It appears that Screenwriter uses a modified RWTS routine of it's own to save files. A small insight into the problem appeared in issue 36 when it was described how to use Screenwriter with the Flash-card. Reference was made that Screenwriter would work with a normal Dos 3.3 RWTS but no description was elaborated on.

I would appreciate hearing from anyone who could help....Thanks.

Marc Batchelor   FL

I thought I'd better pass along some softkeys I have been working on for a while. Since my teacher friend was unable to make reliable backup copies with his bit-copiers, I was called in to remove the protections. It is truly sad that the software manufacturers do not make unprotected copies for teachers to use in the classroom. There are only about two dozen ways that a seven year-old can abuse an original disk (excluding creativity with a hole-punch). That ends up to be costly to the school district. Therefore, I developed these softkeys.

Softkey for ...
Ten Clues
High Wire Logic
Botanical Gardens
A Science Experiment
Tip 'N Flip
Gnee Or Not Gnee
Sunburst Software

Requirements:
A way to reset into the monitor
Demuffin Plus

Sunburst Software uses a highly modified RWTS. However, armed with Demuffin Plus, breaking them is a snap.

1. Format as many blank disks as you have originals.

2. Bload Demuffin Plus at $6000.

3. Boot any one of the DOS 3.3 series listed above and use a NMI card or similar way into the monitor. (I use the Senior Prom).

4. Put Demuffin Plus where it belongs.
803<6000.8103M

5. Start Demuffin Plus
803G

6. Select convert files, and follow the prompts.

Softkey for...
Award Maker Plus
Baudville

For Award Maker Plus, I found the crack to be a two byte change to the code on track $00 sector $05 bytes $FA-FB. Edward Teach's crack was to change these bytes to A9 00. On my version, the code following that change toggles the language card to $C080,X after the A register was transferred to the X register (TAX). With #$00 transfered from the accumulator to the 'X' register this would result in LDA $C080,0 or just $C080. This has the net result of write-protecting the language card. Since the printer driver is loaded into the language card (around $FF00 I think) the program crashes into the monitor as soon as the printer driver was accessed (because the program attempted to load the driver into write-protected RAM. Knowing this, I used Edward Teach's logic (thanks Edward) and changed his crack to A9 01 (instead of A9 00). This allows the driver to be loaded into the RAM card and function normally. I have printed out many certificates using the copy. It functions as well as the original. Please note that this is for my version so I'm not discrediting Edward Teach's crack. If you are uncertain, try both. Just to be on the safe side, print out several certificates in several configurations to be certain that the program does function properly.

Trk	Sct	Byte(s)	From	To
$00	$05	$FA-FB	05 A5	A9 01

Softkey for...
Tag Team Wrestling
Data East

Requirements:
COPYA
Disk Searcher and Sector editor

The first step in cracking this worthless game is to copy it with COPYA modified to ignore DOS errors (B942:18). From there, I boot-code traced the program until I came upon this piece of code:

```
XXBE  20 00 BF   JSR $BF00   Disk check
XXC1  90 03      BCC $XXC6
XXC3  4C DB BE   JMP $BEDB   Crash!
XXC6             Program Start
```

To circumvent the protection was as simple as changing the JSR to $BF00 to clear the carry bit (CLC) and (2) no-operations (NOP). This had the

side benefit of a faster booting disk (no disk check) and a deprotected copy!

1. Boot your DOS 3.3 system disk.

2. Tell DOS to ignore checksum and epilog errors and use COPYA to copy the disk.
POKE 47426,24
RUN COPYA

3. Run your sector editor with search capabilities and search for 20 00 BF 90 03. Change to 18 EA EA 90 03.

4. Write the sector back to disk.

5. (Optional) - Format original and copy since the program is worthless.

Softkey for...
Story Tree
Scholastic Software

Requirements:
A way to capture altered RWTS
SUPER IOB
SWAP controller

This program looked like it was going to be easy to crack, but it had a single surprise. It uses altered RWTS as its sole protection. The crack is not as simple as capturing the RWTS, using the SWAP controller and using your own DOS however, because it crashes into the monitor at the menu. After examining the originals RWTS, Scholastic decided to be cute by putting a RTS at a known location and CALLing it from all the BASIC programs. At first I thought of removing all of the calls. However, I deemed it easier to POKE an RTS to that location at the beginning of the HELLO program (SETUP) instead of trying to locate all of the CALLs.

1. Capture the program's RWTS using whatever means available.

2. Use SUPER IOB SWAP controller to copy the disk to a normal format.

3. Put your own DOS on the disk.

4. Load the program SETUP and add a line at the beginning.
LOAD SETUP
5 POKE 46784,96

5. Save the program back out to the disk.
SAVE SETUP.

That's all on this one.

Softkey for...
Body In Focus
Mindscape

Requirements:
A way to capture altered RWTS
SUPER IOB
File copier
Formatted disks

This one is easy. The only protection is a modified RWTS.

1. Capture the programs RWTS using any means available.

2. Use SUPER IOB SWAP controller to copy the disks to a normal format.

3. Use any file copier to copy the files off of the copies onto DOS 3.3 formatted disks.

Done.

Softkey for...
VCR Companion
Broderbund Software

Requirements:
COPYA
Disk editor and searcher

This disk uses a modified ProDOS (shame shame). It also uses a disk and memory verification routine that resides in memory beginning at $2520. The routines are in the system file VCRC.SYSTEM. Copying the disk is not a large task since it only involves making a change to DOS 3.3 to ignore errors and running COPYA. The resulting copy will not run. Using my Senior Prom, I found the routines. My first attempt was NOP all branches to the spoiler routines. This worked well, but the disk was still being checked (although the outcome did not matter). After examining the routines, I found a way to circumvent the bizarre disk routines and just load the software. Here are some excerpts from the routines:

```
2512: 38        SEC        Error Routine
2513: 60        RTS
2520: 29 0F     AND #$0F    Check the accumulator
2522: C9 04     CMP #$04
2524: D0 EC     BNE $2512   Goto error routine if not
                            good.
2526: AA        TAX
2527: BD 00 60  LDA $6000,X  Check Memory For
                            Tampering
252A: DD 18 25  CMP $2518,X
252D: D0 E3     BNE $2512   ·
252F: CA        DEX
2530: D0 F5     BNE $2527
2559: C6 F5     DEC $F5
255B: F0 5C     BEQ $25B9   If $F5 is zero, crash
255D: 20 DD 25  JSR $25DD   Disk Check Routine
```

2560:	B0 57	BCS $25B9	Crash if error
2562:	A5 F1	LDA $F1	Check for special flag
2564:	C9 07	CMP #$07	
2566:	D0 F1	BNE $2559	If no flag, try disk check again
256A:	BD 8C C0	LDA C08C,X	Another Disk Check
25AB:	BD 8C C0	LDA $C08C,X	
25AE:	10 FB	BPL $25AB	
25B0:	D1 D6	CMP ($F6),Y	
25B2:	D0 05	BNE $25B9	Crash if incorrect byte found!
25B4:	88	DEY	
25B5:	10 F4	BPL $25AB	
25B7:	30 03	BMI $25BC	If ok, branch around next jump
25B9:	4C CC 25	JMP $25CC	Jump to crash
25BC:	A0 F0	LDY #$F0	Disk OK
25BE:	B9 00 02	LDA $0200,Y	
25C1:	99 00 00	STA $0000,Y	
25C4:	C8	INY	
25C5:	D0 F7	BNE $25BE	
25C7:	BD 88 C0	LDA $C088,X	
25CA:	18	CLC	
25CB:	60	RTS	
25CC:	C6 F4	DEC $F4	Disk is a copy
25CE:	F0 03	BEQ $25D3	
25D0:	4C 55 25	JMP $2555	
25D3:	38	SEC	
25D4:	60	RTS	

Seeing the above, it can be figured what my first plan of attack was. I NOPed every branch I could find to $2512 and $25B9. I essentially allowed the program to play with the disk as much as it wanted and pass the check anyway. As it turned out, I had 33 sector edits at first, but it ran (so no insults, ok?). Revising my plan of attack, I changed the code at $2520 (the start of the memory check) to 18 90 0F which clears the carry (CLC) and branches to $2532. My second edit was starting at $255B (start of disk verification) changing to A9 07 85 F1 18 90 5A. This translates to loading the accumulator with #$07 and storing it at $F1 (remember the flag?). I then clear the carry flag (CLC) and branch to the pass routine at the end ($25BC). This bypasses the disk check AND forces the program to store essential bytes in memory.

1. Boot your DOS 3.3 system disk.

2. Tell DOS to ignore checksum and epilog errors and use COPYA to copy all sides of the VCR Companion disks.
POKE 47426,24
RUN COPYA

3. Boot the disk searcher and search for 29 0F C9 04 D0 EC AA. Change the 29 0F C9 to 18 90 0F.

4. Search the same sector for F0 5C 20 DD 25 B0 57. Change to A9 07 85 F1 18 90 5A.

5. Write the sector back to the disk.

6. Copy a new version of ProDOS to this disk.

Softkey for...
Sign Designer
Channelmark

Requirements:
SUPER IOB SWAP
Altered RWTS (see below)
Sector Editor with search capabilities

This Pascal based program gave me fits. In an attempt to figure out the protection, I might have stumbled upon a new form of laziness. Instead of taking the time to create a complex controller, I decided to create my own modified RWTS (if they can, why can't we?) and use the SWAP controller. Talk about lazy. Anyway, aside from altered address and data headers and tailers, there is no other protection. But, once the disk is converted to a normal format, it must be modified to accept normal headers and tailers. Here is how:

1. Move a copy of normal RWTS down to $1900.
1900<B800.BFFFM

2. Modify the heck out of it to read the modified disk:

```
19E7:AA
19F1:D5
19FC:EB
1A3E:AA
1A55:AA
1A5F:D5
1A6A:AB
1A91:DE
1A9B:AB
```

3. Save it as RWTS.SIGN.
BSAVE RWTS.SIGN,A$1900,L$800

4. Run SUPER IOB with SWAP controller installed to copy the disk. Use the "Ignore unreadable sectors" patch to the SWAP controller since sector $00 of track $00 is in a normal format, but the rest of the disk is in a messed up format.

5. Boot up your sector editor and copy track $00 sector $00 from the original to the copy. Track $00, sectors $00, $0C and $0D all look for altered marks.

Trk	Sct	Byte(s)	From	To
00	00	$10	?	D5
		$19	?	AA
		$23	?	96
		$4D	?	D5

	$56	?	AA
	$60	?	AD
0C	$0B	?	D5
	$15	?	AA
	$20	?	96
	$47	?	DE
	$51	?	AA
0D	$9D	?	D5
	$A7	?	AA
	$B2	?	AD
	$EB	?	DE
	$F5	?	AA

That's all of the softkeys I have at the present, but be expecting more in the near future.

Jeffrey A. Wisnia   MA

I sure hope you guys make it. Here's my contribution in return for the countless hours of pleasure you've given me since I became a subscriber in 1982.

I'm enclosing a hardware article which may be of interest to readers with Apple IIe machines.

The Junior Prom

Back when I had an Apple II+, I followed Ernie Young's instructions in COMPUTIST #6 and replaced the F8-ROM in my machine with a 2716 EPROM containing his "reset to the monitor" and page 0-8 moving routines. When I graduated to an Apple IIe (enhanced) I suffered without those helpful features until I modified my antique EPROM burner to program 2764 chips.

This article will show you how to replace the CD-ROM in an enhanced Apple IIe with an EPROM I call the JUNIOR PROM. With it you can reset to the monitor at will or reset to the monitor with memory from $0000-$0900 copied to $2000-$2900 and the stack pointer copied to $2901. Memory from $2000-$3FFF is the Hires screen area, which usually isn't part of the permanent program code in Hires games, so you can usually capture all the code from single load programs in one shot.

The memory copy feature lets you examine the program code located at $400-$7FF, which otherwise gets overwritten by text on the screen as soon as you enter the monitor, as well as page zero, the stack, and the page two keyboard buffer, areas which get altered by a reset to the monitor or by entering commands from the keyboard. The ability to copy these pages where they don't get altered is very helpful for general software snooping and copy protection analysis. The memory copy routine itself is lifted directly from the original Ernie Young's original article.

In addition, the IIe's nasty habit of blasting two bytes in every page of RAM on a reboot is exorcised in the altered ROM.

You will need a 2764 EPROM and access to an EPROM programmer which can burn the contents of an 8K long binary file into the 2764. These EPROMs are presently available for under $5 each from most electronic parts suppliers catering to individuals, check the ads in any hobby electronic magazine.

The Junior Prom will leave your Apple IIe without cassette tape storage routines, but you won't miss these dinosaurs. Applesoft BASIC under ProDOS does not even support tape storage and issues a SYNTAX ERROR in response to SAVE and LOAD commands. To keep things neat, my code returns an ERR message for monitor tape commands W and R, and Applesoft Basic's SAVE and LOAD under DOS 3.3.

Note that the JUNIOR PROM won't work in an unenhanced IIe, but by now you should have upgraded to the enhanced version anyway, to get the faster 80 column routines and Mousetext.

First, you must capture the code of the CD-ROM in your IIe by moving it down in RAM. You can't BSAVE it directly to disk because it occupies the same memory area as the disk controller, which gets switched off when the CX-ROM is switched in. My EPROM programmer wants its data loaded at $2000, so that's where I chose to move it.

To read the CX-ROM from the monitor you have to tickle $C007 with a write command in order to switch it into the $C100-$CFFF space normally occupied by peripheral card ROM. The CD-ROM in your IIe contains zeros from $C000-$C0FF, but this range of the ROM is never switched in, as that memory range is reserved for the Apple's I/O softswitches, annunciators, game paddles and such.

Enter the monitor:

```
CALL -151
C007:00                switch in CX-ROM
2100<C100.DFFFM        copy memory
2000:00 N 2001<2000.20FEM zeros C000-C0FF
C006:00                switch the slot ROMS back in
BSAVE CD.ROM, A$2000, L$2000  if you mess
up later, you can reload this
```

You now have an image of the CD-ROM in memory. Enter the following hexdumps from the monitor, changing the high byte of the addresses if you saved the CD-ROM code at a starting location other than $2000:

PART1

```
22C0: 4C D0 C2 AD 61 C0 10 1A   $8496
22C8: A9 00 8D F4 03 4C E2 C2   $3A7C
22D0: 2C 00 C0 10 FB AD 10 C0   $9045
22D8: AD 00 C0 4C AD C5 EA EA   $F2D8
22E0: EA EA                     $4123
```

PART2

```
2598: 20 2D FF 20 62 FC 68 68   $0DE8
25A0: A9 FE 48 A9 C4 48 60 00   $CAB6
25A8: 00 00 4C 98 C5 C9 15 F0   $03E8
25B0: 07 C9 4D F0 3E 4C 00 C6   $A09B
25B8: BA 8E 01 29 A0 00 B9 00   $CCBC
25C0: 00 99 00 20 B9 00 01 99   $022A
25C8: 00 21 C8 D0 F1 4C D4 C5   $20F5
25D0: EA 4C 98 C5 84 3C 84 42   $4AAA
25D8: 84 3E A9 09 85 3F A9 02   $3818
25E0: 85 3D A9 22 85 43 20 2C   $AC69
25E8: FE A9 0D 8D D0 07 A9 3E   $021B
25F0: 8D D1 07 A9 FF 48 A9 58   $BC03
25F8: 48 4C C5 FE FF D4 00 00   $4DE8
```

Save the JUNIOR PROM code:
BSAVE JR.PROM, A$2000, L$2000

As a check that you didn't accidently change anything elsewhere in the code, compare it with the original:
BLOAD CD.ROM, A$4000
2000<4000.5FFFV   *monitor verify command*

You should see differences in the changed areas $22C0-22E1 and $2598-$25FF only. Any other differences mean you messed up somewhere and should start over.

Program a 2764 with the code you saved as JR.PROM. Turn off your Apple, pull the CD-ROM and install the JR.PROM. Check the orientation of the EPROM making sure the notched or dotted end points toward the keyboard, the same as the adjacent chips. A reversed chip will be destroyed the instant you turn on your Apple, and might take out something else in the process too!

Using the JR.PROM will make no difference in the normal operation of your computer, except for the loss of cassette tape functions. The two byte ROM checksum was made the same as the original CD-ROM, so it will pass ROM checksum tests made by some copy protected software.

To use the added features of the JR.PROM, press the control, closed Apple and reset keys. (Press reset after the other two keys are down.) Your Apple IIe should freeze, awaiting the next keypress. What happens next depends on which key you hit:

Press M (upper case M) to drop into the monitor.

Press -> (right arrow) to drop into the monitor with the memory copy and stack pointer saves described above. An inverse "M>" will print in the lower left corner of the screen to indicate that memory copy was done.

Pressing any other key will start the Apple's diagnostic routines.

Disassemblies of the new code:

The start of the original reset routine is $C2B0.

```
C2B0 A9 FF      LDA #$FF
C2B2 8D FB 04   STA $04FB   blast mode byte
C2B5 AD 5D C0   LDA $C05D   reset annunciators
C2B8 AD 5F C0   LDA $C05F
C2BB AD 62 C0   LDA $C062   test closed apple key
C2BE 10 03      BPL $C2C3   not pressed, try open
                            apple
```

The above code is unchanged in Junior Prom, and is shown for reference.

```
C2C0 4C D0 C2   JMP $C2D0   pressed, get next key
C2C3 AD 61 C0   LDA $C061   test open apple key
C2C6 10 1A      BPL $C2E2   not pressed, goto reset
C2C8 A9 00      LDA #$00    pressed
C2CA 8D F4 03   STA $03F4   destroy powerup byte
C2CD 4C E2 C2   JMP $C2E2   jump to reset (will boot)
C2D0 2C 00 C0   BIT $C000   test keyboard
C2D3 10 FB      BPL $C2D0   wait for keypress
C2D5 AD 10 C0   LDA $C010   clear keystrobe
C2D8 AD 00 C0   LDA $C000   get key value
C2DB 4C AD C5   JMP $C5AD   goto rest of new routine
C2DE EA         NOP
C2DF EA         NOP
C2E0 EA         NOP
C2E1 EA         NOP
```

New return from tape read or write calls is at $C598.

```
C598 20 2D FF   JSR $FF2D   monitor print "err"
C59B 20 62 FC   JSR $FC62   monitor print C/R
C59E 68         PLA         pull return address
C595 68         PLA
C5A0 A9 FE      LDA #$FE    push return address
C5A2 48         PHA
C5A3 A9 C4      LDA #$C4
C5A5 48         PHA
C5A6 60         RTS         jump to $FEC5 via RTS
C5A7 00         BRK         Note: $FCE5 turns off
C5A8 00         BRK         CX-ROM and then does
C5A9 00         BRK         another RTS
```

Old tape write entry point is $C5AA. It is called from EF-ROM, so we must use it too.

```
C5AA 4C 98 C5   JMP $C598   goto new return routine
```

Keypress tests begin here at $C5AD.

```
C5AD C9 15      CMP #$15    right arrow key?
C5AF F0 07      BEQ $C5B8   Yes, goto memory copy
C5B1 C9 4D      CMP #$4D    "M" key?
C5B3 F0 3E      BEQ $C5F3   Yes, goto exit code
C5B5 4C 00 C6   JMP $C600   Neither, jump to
                            diagnostics
C5B8 BA         TSX         start of Young's routines
C5B9 8E 01 29   STX $2901   Save stack pointer
C5BC A0 00      LDY #$00
```

```
C5BE B9 00 00   LDA $0000,Y  copy first two pages
C5C1 99 00 20   STA $2000,Y  this way to avoid
C5C4 B9 00 01   LDA $0100,Y  changing any
C5C7 99 00 21   STA $2100,Y  memory in them
C5CA C8         INY
C5CB D0 F1      BNE $25BE   loop till copy completed
C5CD 4C D4 C5   JMP $C5D4   jump over old tape read
                            at $25D1
```

Another old tape read entry point is $C5D1. It is called from EF-ROM, so we must use it too.

```
C5D1 4C 98 C5   JMP $C598   goto new return routine
```

Continue with new code at $C5D4.

```
C5D4 84 3C      STY $3C     Setup for monitor move
C5D6 84 42      STY $42     routine, (Y=0 at entry)
C5D8 84 3E      STY $3E
C5DA A9 09      LDA #$09
C5DC 85 3F      STA $3F
C5DE A9 02      LDA #$02
C5E0 85 3D      STA $3D
C5E2 A9 22      LDA #$22
C5E4 85 43      STA $43
C5E6 20 2C FE   JSR $FE2C   monitor memory move
C5E9 A9 0D      LDA #$0D    screen inverse "M"
C5EB 8D D0 07   STA $07D0   store lower left screen
C5EE A9 3E      LDA #$3E    screen inverse ">"
C5F0 8D D1 07   STA $07D1   store next to "M"
C5F3 A9 FF      LDA #$FF    push monitor entry return
C5F5 48         PHA         address onto stack
C5F6 A9 58      LDA #$58
C5F8 48         PHA
C5F9 4C C5 FE   JMP $FEC5   jump to $FEC5
C5FC FF         ???         these two bytes make the
C5FD D5         ???         ROM checksum come
C5FE 00         BRK         out right
C5FF 00         BRK
```

Note: The Applesoft SAVE and LOAD commands produce double "ERR" messages when running under DOS 3.3 or with DOS disconnected. This is because these Applesoft commands each make two calls to the monitor tape WRITE and READ routines before they are done. Eliminating this minor glitch looked like it would take a lot more code or a change to the EF-ROM and didn't seem worth it.

Bob Igo   PA

Enhancements to APT Scanner

Just add or modify the following lines in the BASIC program and type in the new binary file "SCANNER.AI" below and save it and you will have what I call an "artificial intelligence" option. The program chooses the most likely locations for the necessary APT information by checking after each occurrence of LDX #$xx, LDY #$xx or LDA #$xx to see if there is a store command. If not, it doesn't tell you. In case it eliminates a necessary location, I have the option of using no screening.

APT Scanner Patches
LOAD APT.SCANNER
```
95 IF PEEK (768) < > 169 THEN PRINT
   CHR$ (4) "BLOADDOS~UTILITY"
100 HOME : PRINT "WOULD~YOU~LIKE~ME
   ~TO~USE~ MY~INTELLIGENCEAND~FOCUS
   ~ONLY~ON~THEMOST~LIKELY~~~~~~~
   CHOICES?"
105 GET G$: PRINT G$: IF NOT (G$ = "Y"
   OR G$ = "N" ) THEN CALL - 198: GOTO
   100
106 IF G$ = "Y" THEN PRINT CHR$ (4)
   "BLOAD~SCANNER.AI" : GOTO 110
107 PRINT CHR$ (4) "BLOAD~SCANNER"
```
SAVE APT SCANNER
Enter the new scanner assembly program and save it.

SCANNER.AI
```
6000: A9 A9 8D 00 61 A9 A2 8D   $8439
6008: 01 61 A9 A0 8D 02 61 A0   $6F86
6010: 00 02 80 B9 00 20 DD 00   $5A68
6018: 61 F0 09 E8 E0 03 D0 F3   $7E10
6020: C8 D0 EE 60 B9 01 20 CD   $E1A7
6028: 50 61 D0 EF 8E 03 61 A2   $4D89
6030: 0D BD 69 60 D9 02 20 F0   $E0A1
6038: 09 CA D0 F5 AE 03 61 4C   $A5C7
6040: 1B 60 A9 00 85 24 AE 03   $E4CB
6048: 61 AD 10 03 20 DA FD A9   $6F83
6050: 10 85 24 AD 15 03 20 DA   $C02E
6058: FD 20 85 25 AD 20 98 20 DA $5ECA
6060: FD 20 8E FD AE 03 61 4C   $B653
6068: 1B 60 85 95 8D 9D 99 91   $66C9
6070: 81 86 96 8E 84 94 8C      $58C7
```
BSAVE SCANNER.AI, A$6000, L$77

Scanner.AI Disassembly
```
A9 A9 LDA    #$A9
8D 00 61 STA  $6100
A9 A2 LDA    #$A2
8D 01 61 STA  $6101
A9 A0 LDA    #$A0
8D 02 61 STA  $6102
A0 00 LDY    #$00
A2 00 LDX    #$00
B9 00 20 LDA  $2000,Y
DD 00 61 CMP  $6100,X
F0 09 BEQ    $6024
E8 INX
E0 03 CPX    #$03
D0 F3 BNE    $6013
C8 INY
D0 EE BNE    $6011
60 RTS
B9 01 20 LDA  $2001,Y
```

```
CD 50 61   CMP $6150
D0 EF BNE  $601B
8E 03 61   STX $6103   New stuff
A2 0D LDX  #$0D
BD 69 60   LDA $6069,X
D9 02 20   CMP $2002,Y
F0 09 BEQ  $6042
CA   DEX
D0 F5 BNE  $6031
AE 03 61   LDX $6103
4C 1B 60   JMP $601B
A0 00 LDA  #$00
85 24 STA  $24
AE 03 61   LDX $6103   New
AD 10 03   LDA $0310
20 DA FD   JSR $FDDA
A9 10 LDA  #$10
85 24 STA  $24
AD 15 03   LDA $0315
20 DA FD   JSR $FDDA
A9 20 LDA  #$20
85 24 STA  $24
98   TYA
20 DA FD   JSR $FDDA
20 8E FD   JSR $FD8E
AE 03 61   LDX $6103   New
4C 1B 60   JMP $601B
85 95 8D 9D 99 91 81 86 96 8E 84 94 8C   Opcodes for
                            store commands
```

Using this program, I've come up with some more APT's.

Advanced Playing Technique for...
Trolls and Tribulations:
?

Unlimited Lives
Trk	Sct	Byte(s)	From	To
08	08	CB-CC	C6 13	EA EA

Advanced Playing Technique for...
Microwave
?

Unlimited Lives
Trk	Sct	Byte(s)	From	To
11	0D	3E-40	CE BD 0B	EA EA EA
19	0A	42-44	CE BD 0B	EA EA EA

Advanced Playing Technique for...
Oil's Well
Sierra On Line

Extended Lives
Trk	Sct	Byte(s)	From	To
17	07	5C	03	FF (0-FF)

How to use APT SCANNER

As Mr. Polasky requested, here is an explanation of my technique with regards to Oil's Well. First, let me say that I tried the timer but was unsuccessful. I know that the number 99 ($63) is stored at location $6927 on track $17, sector $0C (it was not necessary to search for something like 990 because the time goes down in factors of ten), but it isn't being decremented at all. The program is apparently using some sort of unorthodox way of taking the value in $6927 and storing it somewhere else, messing with it and decreasing it. I've scanned memory for the time at various points but to no avail. Help!! Someone else out there must be able to defeat this.

Now, for the lives feature. I was unable to make unlimited lives because of the same odd schemes I described above, but I managed to extend them. I read that Frank desired a little APT help. I needed an example to use with my updated APT techniques. In numbered steps, here is what I did:

1. I booted up the game and saw that three lives were displayed on the screen. This meant that I would need to scan for 03 with the APT SCANNER.

2. I booted up APT SCANNER and chose the new "Artificial Intelligence" option. I chose 0 as the starting track and 34 as the ending track. I used slot 1 for my printer, but slot 0 is OK for the monitor. After the SCANNER was done, it had told me the following information:

TRACK	SECTOR	BYTE
00	0D	A6
01	03	90
01	05	8C
10	0B	96
15	00	01
15	02	DC
16	02	3D
17	06	DF
17	06	EF
17	07	5B
17	09	61
17	0F	1D

This means that in all these places, 03 is being stored in a certain memory location. Although a fully independent version of APT SCANNER is on the horizon, we must for now use a sector-editor with disassembly capabilities, such as Copy II+, for the next step.

3. Boot up Copy II+ and get into the sector-editor. Read each one of the above locations (although tracks 0-2 can usually be ignored, as this is where the DOS resides most of the time).

For example, the first one I read was track $10, sector $0B. I jumped to byte $96 and used the (L)ist option. It told me that 03 was being stored somewhere in DOS. I figured that it couldn't be the number of lives (it turned out later that I figured correctly). So, I read the rest of the locations and found the following storing addresses:

Trk	Sct	Byte	Storing location (In Hex)
15	00	01	0363, 0364
15	02	DC	0376
16	02	3D	8429
17	06	DF	0353
17	06	EF	0363, 0364
17	07	5B	0225, 0226, 0385, 902A
17	09	61	0229,X
17	0F	1D	06B0,X

4. Now, boot up the game and get "killed" once. You should currently have two lives. Break the program's execution. If you can't do this without making the program reboot itself, use CTRL-OPEN APPLE-RESET and then immediately after the computer reboots, hit CTRL-RESET. It may mess up a bit of memory, so you might want to try searching the disk for F2 03 and see what the program is doing to the reset vector ($3F4-$3F4). Look one byte before F2 and list. If it is storing anything at 3F2, 3F3, or 3F4, just NOP the entire area. This should enable a safe CTRL-RESET. Anyway, the idea here is to look at all the memory locations above (storing locations) and see if any of them have 02, our current number of lives, in them. The reason I think it's best to get "killed" once before looking at the locations is because it is more likely that 03 is going to be in a location that isn't the number of lives because we know that 03 is being stored in every location above at least once. The chance of finding an 02 after we lose one life is less probable and therefore more accurate. If you have a multiple player option, choose it and lose one life per player. This also helps to make things more accurate. I looked at all the above locations after getting killed once. I found 02 to be in location $225. Because this is a two-player game, I checked again for 02 and found it in location $226. I already told you about the trouble I had with making an unlimited lives feature, so I'll just tell you what I did in this case. If you want to know how to make "unlimited" options in other cases, read Computist #65, page 17, under steps 5 and 6.

5. I tried changing byte $5C on track $17, sector $07 from $03 (for three lives) to $FF (which should make most people happy). I booted the game, and aside from a large amount of little munchie-looking things which display the current number of lives, the game functions fine.

So, that's how I did it. No magic or anything. If someone does know how to make this option unlimited, please let me know how you did it.

Bugs

There were a couple of small errors in my DeathSword APT article that were pointed out to me. They might be a bit confusing for some, so I'll tell you what they should be. I said under section 2 that DEATH.SWORD began on track $05, sector $09 and ended on track $09, sector $06. That is correct. However, under section 3, I said that "I read track $09, sector $06 which is where the file DEATH.SWORD starts..." This should have read "track $05, sector $09" instead. Also, under section 5 (now, this was a half typo, half temporary brain damage error) I stated that the number, $1234 was a 4-byte number. Actually, it's a 2-byte number, and $4B is a 1-byte number. Sorry about any confusion I may have caused.

Mopman   MA

I received the urgent message and I am sorry to hear that Computist is going to reduce the number of issues. Before you receive this you should already have my check for my membership. In spite of a high price, I feel that I have gotten more than my money's worth from the magazine. I have not been a big one on keeping notes so a lot of cracks that were simple or very similar to published ones have been lost forever. But here are some that were unusual enough for me to keep good notes on. In the future I will try to be more active in submitting softkeys for all cracks. Two of them involve the use of Copy 2+'s more advanced parameters. On the subject of parameters Copy 2+ has a lot of undocumented parameters that they say are used "to control very specific copy methods for certain protection schemes". Does anybody have any information on these? I really prefer to make my programs COPYA-able. But I guess the main objective is to make a backup by whatever means available.

Softkey for...

Dinosaur Days
Pelican Software

This softkey requires that you be reasonably familiar with the use of the nibble editing feature of COPY 2+. When completed you will have a utility disk that will copy and deprotect Dino Days and some other Pelican software. When viewed with a nibble editor DINO DOS looks like this when compared to a normal DOS 3.3 address header.

Normal

```
          D5 AA 96 FF FE AA
AA AA AA FF FE DE AA EB
```

Dino DOS

```
D5 FF FE D5 FF FE xx xx D5 FF FE AA
AA AA FF FE FF FF
```

D5 FF FE repeated twice, one or two odd bytes then D5 FF FE, followed by the track, sector, checksum and epilog. The epilogs have been changed to FF FF.

Capturing the RWTS and using a swap controller would not produce a working copy. I suppose I could have BOOT TRACED but I never really had much luck at it. I am pretty good with a nibble editor, so I produced my first copy by changing the address header on all the sectors before the track was written to disk. After doing all this I found that DINO DOS has the ability to read their protected format, or read and write normal DOS 3.3. So you only have to be concerned with capturing the first three tracks. Track 0 has only the address epilogs changed so most of the work has to be done on track 1 and 2.

Boot Copy 2+ and format a blank disk (DOS 3.3). Get to the Bit Copy and Sector Copy menu. Select Sector Edit and press "P", patched. Then choose DOS 3.3 PATCHED. Escape out of this mode and select Sector Copy. Copy only Track 0 from your original to your formatted disk. Then select Manual Bit Copy. When asked, Start Track 1, End Track 2, Sync No, Keep Length No. Press / to modify parameters, what parm. 0B to what value 02. Changing 0B to 02 puts you in the nibble editor before copying to your destination disk. So press return to begin copy. When the editor displays the track buffer you probably will not be near sector 0. Hold down the "T" key to scan back to the beginning of the buffer. When it beeps you are at the beginning. Press "Control B" to set this as the beginning. Press "F", Find, then type AA AA and press return. You should see: D5 FF FE D5 FF FE xx xx D5 FF FE AA AB AA AA FF FF FF FF. You should be on the first AA of the AA AA bytes. Use the "J" key to move 7 places to the left. Type "C" to change the bytes to D5 AA 96 then press return. Then use "J" to move you back to the D5 you just entered, and press "Control B" to reset the track beginning. Type "F", Find, and type D5 FF FE and press return. Press "R", Repeat Find, twice and you should be positioned on D5. Press "J" twice and press "C", and type D5 AA 96 again. Repeat this for the remaining sectors. When you do the 16th, sector do a "R" one last time. Then look back about 4 lines and find DE AA EB. Using the I, J, K, M keys, position the cursor on the byte after EB. Press "Control E" to set the track end. In the upper right corner of the screen you should see LENGTH, and it should read approximately 17DF bytes. Scan through the sectors and check to make sure you did not miss any. When complete press "Q" to quit and write to the destination disk. Repeat procedure for track 2.

Again use the sector editor and select DOS 3.3 PATCHED. Read the sectors on track 1 and 2 to make sure they are all readable. If some are not readable repeat the procedure until they are. Selecting DOS 3.3 Patched on the sector edit menu also sets this parameter for Sector Copy. Use your newly created disk as the source disk and use Sector Copy to recopy it. This fixes the bad address checksums. Boot your System Master disk and enter the monitor, CALL-151. Tell DOS to ignore address epilogs B993:00 B99D:00. Reenter basic, 3D0G, and run Copy A. This gives you a disk with DOS that can handle their protection or normal disks. On this disk you can add a copy program such as Super IOB with a standard controller, Copy A, FID, or most any that does not have it's own DOS. The current booting program of DINO DOS is called HI, so you need to change it, or change the name of the copy program you are using. The utility disk you have made will also copy Limited Edition Mini Converter by Pelican.

Bitkey for...

Story Tree
Scholastic
Magic Spells
Learning Co.

This details how to make a bit copy of these Scholastic titles and also quite a few other programs using similar protection.

In general track 0 is normal. Most of the disk will have altered Epilogs and usually 2 or 3 tracks will have only 1 or 2 areas of valid data. Each of these areas will contain 5 sectors, 3 4 5 6 7, A B C D E, 1 2 8 9 F they may be written on full tracks, 2, 3 or half tracks 2, 2.5, 3 or any other combination of quarter, half or full tracks. Because the tracks can overlap it is very important to set the copy program not to erase the track before writing to it. They normally write $782 bytes per group of data. Between the groups of data is invalid bytes not sync bytes. For this reason it is very important that you start with a new disk or erase a disk with a magnet for use as a destination disk.

Its best if you write a Copy 2+ parameter to do the copying, because to do a different program, all you may have to do is alter the track numbers. This are the parameters for Story Tree.

T0-T2, 55=03
T3,0D=01,44=01,4D=00,55=03,45=07,46=82, 15=AA, 16=AB, 29=AA, 2A=AB, 2D=FF, 2E=FF
T3.5, 0D=01, 44=01, 4D=00, 55=03, 45=07, 46=82, 15=AB, 16=AE, 29=AB, 2A=AE, 2D=FF, 2E=FF
T4,0D=01,44=01,4D=00,55=03,45=07,46=82, 15=AF, 16=AB, 29=AF, 2A=FF, 2D=FF, 2E=FF
T5-T11,4D=01,44=00,0D=00,55=03,15=AA, 16=AA

The parameter for Magic Spells is the same except that in the last line change T5-T11 to T5-T22.

For tracks 0 to 2, the parameter finds the track start by header. For track 3, they set the sector to AA AB, this is sector 1, find track start by header, sync. to sector 1 on track 0, do not erase the track, and cut the track to $782 bytes.

Track 3.5 is the same except we set the sector to AB AE sector 6

Track 4 we set the sector to AF AB sector B.

Track 5-11 everything is restored to normal.

Softkey for...

Case of the Great Train Robbery
Mystery of the Witches Shoes
Case of the Missing Chick
Troll Associates

All 3 of these programs use the same protection. A nibble count on track 3 and a signature check on track 23. But we can easily make these COPYA-able. Use a copy program that will ignore the read error on track 3. Such as Copy Disk from the Copy 2+ menu. Copy your favorite DOS to the disk and change the Boot program to "MENU". On Case of the Missing Chick, change the Boot program to "U". That's all.

On other programs, use View Files to find the file that BRUN's a lot of other programs and change it to be the Boot program.

Softkey for...

Science 4
Understanding our Solar System
The Ellen Nelson Learning Library

This is another one of those sneaky signature checks. But it is easy to fix. Boot your System Master disk. Enter the monitor, CALL-151, tell DOS to ignore Address Epilogs B993:0 B99D:0 and ignore Data Epilogs B925:18 60 and return to BASIC, 3D0G. Run COPYA and when done use your sector editor to edit track 0 sector 5. Change byte position $44 from $A9 to $60.

Paul A. Johnson   IA

To George Cawthorne: Here are two ways to change from a "Block, Address, Change-to" format to a "Track, Sector, Address, Change-to" format. The first method is to edit Super IOB 1.5 to automatically perform the conversion while making sector edits (lines 310-340), and the second method is a program that asks for the block and address and gives the corresponding track, sector and address.

When using the second method, you have to run the program and jot down the conversions next to the block edits in the issue of Computist. Then, when it says to make block edits, make sector edits instead, using your favorite sector editor (I use Copy 2 Plus).

The first method needs a little more explanation. The program "SIOB.BLOCK.FIX" contains only one line. Add this to Super IOB by (1) loading SUPER IOB, and (2) typing the line, and (3) saving Super IOB. Once SUPER IOB is modified, when you want to make block edits instead of sector edits, use the following syntax in the data statements:

DATA 3 BCHANGES, Block #, 0, Address #, Change-to

The space after the 3 is necessary, and there cannot be a space between the B and CHANGES. The second data item of each block edit is not used, so it can be any number. The variable that contains that number after the read will have the sector number after line 335 is executed.

I have included a sample controller, which uses the information about Moonglum, which was in issue #69. If you have a disk that does not need to be copied with Super IOB, (it is copyable with a fast-copier), then the controller should have the following:

TK = First track, sector to be edited
LT = Last track, sector to be edited (minus 1 MOD 16)
FAST=1

Or, if you'd rather, copy the entire disk by using TK=0:LT=35:ST=15:LS=15. Re-reading the Super IOB documentation concerning R/W a range and Sector Edits might be helpful.

Now, for some suggestions, questions, and stuff.

Suggestion #1: I noticed that issue #66 of Computist is not available as a back issue. I assume that it is too difficult to make photocopies of the tabloid format. If an annual book isn't in the future for these tabloid issues, is it possible to put the issue on disk as text files that the reader can load into their favorite word processor and print out a copy? I don't know how much disk

---

space this would take, but, even if it took more than one disk, that would be better than nothing for the people who want that information. This suggestion means nothing (of course) if an annual "review" is going to be printed. Then the person could just buy the book.

Suggestion #2: (to the people who write to the RDEX) Is it possible to say a couple of words about the software you are deprotecting. I am not asking for an "all-out" review (we don't want to take Jeff Hurlburt's job), just a couple of simple sentences like "A Role-Playing Game set in future times. Good Game, worth the money". Quite a few times I've read a softkey in Computist and I can't tell if the software is a game, educational program, or a utility of some type. Even if I can figure out a category for the software, I don't know if it is worth 2 cents or not.

⑦ This has nothing to do with computers (at least not directly), but every time I have a question, someone who reads Computist has the answer. I recently determined that my house does not have grounded wiring. The outlets are 3-pronged, but only two wires go to the outlet; the ground wire is missing. I need to somehow ground the outlet that my computer uses, but the wiring goes up to the attic (which is not accessible), and then down to the fuse box. We have a finished basement, so I cannot just run a grounded wire directly. Now for my question. Could I use a GFI (ground fault interrupt) outlet, like the ones in the bathroom. The diagram on the package says it is possible, but I thought I'd better ask someone who knows more about electricity than I do. Another thought that I had was to run a wire from the ground post on the outlet to something that is grounded, like the plumbing pipes. Is this a good idea, or am I just asking for trouble? Any help I can get is greatly appreciated, and the sooner I can ground my computer, the sooner I can get a good nights' sleep.

Free software from Beagle Bros

Stuff #1: Just in case anybody is interested. Beagle Brothers is giving away free software. All you have to do is call up Pro-Beagle, the Beagle Brothers BB. You can download many of the Beagle Brothers older programs, like Beagle BASIC, DOS Boss, Fat Cat, Pro-Byter, and more! The programs on the BB are free, you can even distribute them to your friends. The number for Pro-Beagle is (619) 558-6151. To most people, this is a long distance call, but if you have a fast modem, you can download all of the programs for about the cost of one or two of them at a discount software store.

SIOB Block Patch

```
335 IF RIGHT$ (A$, 8) = "BCHANGES" THEN
    A5 = INT (A1 / 8):A6 = (A1 - A5 *
    8) * 4:A2 = A6 - 15 * (A6 > =
    15):A2 = A2 + 2 * (A3 > 256):A3 =
    A3 - 256 * (A3 > 256):A1 = A5
```

SAMPLE.BLK.CON

```
510 REM THIS DELETES LINE 510 IN SUPER
    IOB
1000 REM SAMPLE CONTROLLER USING BLOCK
     EDIT
1010 TK = 25:LT = 26:ST = 15:LS =
     15:CD = WR:FAST = 1
1020 GOSUB 490: GOSUB 610
1030 T1 = TK:TK = PEEK (TRK) - 1:
     GOSUB 310:TK = T1
1040 GOSUB 490: GOSUB 610: IF PEEK
     (TRK) = LT THEN 1060
1050 TK = PEEK (TRK):ST = PEEK (SCT):
     GOTO 1020
5000 DATA 3~BCHANGES
5010 DATA 205,0,442,36
5020 DATA 205,0,447,36
5030 DATA 205,0,452,128
5040 REM BLK 205 -> TRK 25, SCT 5 & 7
5050 REM ADD 442 -> SCT 7, ADDR 186
5060 REM 447 -> SCT 7, ADDR 191
5070 REM 452 -> SCT 7, ADDR 196
5080 REM LINE 510 DELETES LINE 510 IN
     SUPER IOB. THIS ALLOWS COPYING FROM
     DRIVE 1 TO DRIVE 1 WITHOUT A
     MESSAGE ASKING TO CHANGE DISKS. IF
     YOU ONLY HAVE ONE DRIVE, THEN YOU
     NEED TO RELOAD SUPER IOB IN ORDER
     TO COPY ANOTHER DISK.
```

Checksums

510-$684E	1050-$1BE9	5050-$4E7C
1000-$232A	5000-$9858	5060-$71AE
1010-$683D	5010-$B726	5070-$3DB8
1020-$6DB6	5020-$EBE7	5080-$9D71
1030-$1685	5030-$3900	
1040-$4C82	5040-$952F	

BLOCK.2.SECTORS

```
10 HOME :HX$ = "0123456789ABCDEF"
20 GOSUB 1000
30 TRK = INT (BLK / 8)
40 SEC = (BLK - TRK * 8) * 4 - 15 *
   ((BLK - TRK * 8) * 4 > = 15)
45 IF BY > = 256 THEN BY = BY -
   256:SEC = SEC + 2
50 H = TRK: GOSUB 3000:TRK$ = H$
60 H = BY: GOSUB 3000:B1$ = H$
80 SK$ = MID$ (HX$, SEC + 1,1)
100 PRINT : PRINT "Block-$" BLK$
    ",~Byte-$" BY$ "~=~Track~$" TRK$
    ",~Sector-$0" SK$ ",~Byte-$" B1$
110 PRINT : PRINT
    "Any~Key~to~Continue" : WAIT -
    16384,128: POKE - 16368,0: GOTO 10
1000 INPUT "Block-#~(in~Hex)~$" ;BLK$
1005 IF LEN (BLK$) = 0 THEN END
1010 A$ = BLK$: GOSUB 2000:BLK = A
```

```
1020 INPUT "Byte~#~(in-Hex)~$" ;BY$
1030 IF LEN (BY$) = 0 THEN HOME : GOTO
     1000
1040 A$ = BY$: GOSUB 2000:BY = A
1045 PRINT "BY=" BY "~BLK=" BLK
1050 RETURN
2000 A = 0: FOR X = LEN (A$) TO 1 STEP
     - 1
2010 FOR Y = 0 TO 15
2020 IF MID$ (A$, LEN (A$) - X + 1,1)
     = MID$ (HX$,Y + 1,1) THEN A = A + Y
     * 16 ^ (X - 1):Y = 15
2030 NEXT Y,X
2040 RETURN
3000 H$ = MID$ (HX$, INT (H / 16) +
     1,1) + MID$ (HX$,H - INT (H / 16) *
     16 + 1,1): RETURN
```

Checksums

10-$4DCD	100-$88A3	1045-$115A
20-$15E0	110-$1AFE	1050-$6984
30-$C681	1000-$94D8	2000-$7A60
40-$1F5E	1005-$50CC	2010-$2098
45-$CBCF	1010-$0733	2020-$ACEB
50-$02B7	1020-$4DF2	2030-$4880
60-$F409	1030-$0EB2	2040-$9CBB
80-$4D8F	1040-$3447	3000-$DF96

Advanced Playing Technique for...

2400 A.D.

Origin

The Upper Floors

To Dave Stewart: In Computist #67, you asked for the information to map the upper floors of 2400 A.D. Actually, when I was done with the game, I gave all of my material to a friend so he could play the game, and he returned the game when he was finished, but not any of the other information. I decided that I could dig through the file again and come up with the information, and to tell the truth, it was easier the second time.

Requirements:

Computist #63 (Original 2400 A.D. Mapper)
2400 A.D. Character Disk

1. Load (or type in) the original 2400 A.D. Mapper

2. Add these modified or extra lines.

LINES.2.MODIFY

```
100 DIM BY(136)
110 INPUT "WHICH~PAGE-(1-18):" ;PG
1000 RESTORE : FOR B = 1 TO 136: READ
     BY(B): NEXT : RETURN : REM : PRINT
     CHR$ (4) "PR#1" : RETURN
1010 DATA 30,72,29,28,31,72,72,72,23,
     22,72,21,72,20,0,1,7,6,8,9,15,14,16,17,72,
     32,72,33,72,34,72,35,25,27,24,26,72,72,
     2,3,5,4,10,11,13,12,18,19,72,38,36,37,72,
     39,72
1015 DATA 53,72,72,72,52,72,72,72,72,
     54,72,55,72,72,72,72,46,47,72,42,72,41,
     45,72,72,72,72,72,72,56,72,57,48,49,
     43,72,40,72,72,44,72,50,72,51,72,72,
     72,72
1016 DATA 61,62,72,60,72,72,66,72,63,
     64,59,72,58,72,72,65,69,72,72,68,72,72,
     71,72,72,70,72,72,67,72,72,72
```

Checksums

100-$E92E	1000-$8398	1015-$5015
110-$2137	1010-$9034	1016-$59B3

3. Save the modified program.

Add the following figures under Figure 1 in the article in Computist #63:

Figure 1A: Metropolis (2nd Floor)

```
|        |        |         |         |
| Page 9 |        |         |         |
| Page 8 | Page 10 | Page 12 | Page 13 |
|        | Page 11 |         |         |
|        |        |         |         |
```

Figure 1B: Metropolis (3rd Floor)

```
|        |         |         |
|        |         |         |
|        | Page 14 | Page 15 |
|        |         |         |
```

Figure 1C: Metropolis (4th & 5th Floors)

```
|        |         |         |
|        | Page 16 | Page 17 |
|        |         |         |
```

I once again saved some paper by moving three "areas". The lower left corner of page 14 is the 3rd floor of the University Library, the lower right corner of page 15 is the 3rd floor of the Novue Apartments, and the lower left corner of page 16 is the 5th floor of the Tzorg Authority Complex. Note: the same symbols from the original program are used, so the "vending machines" in the Tzorg Authority Complex are not going to show up on the map. If you wanted, you could determine the numbers for the "vending machines" and add them to the numbers in line 210. OK, OK, OK, stop twisting my arm, the numbers are $14, $15, $16, & $17 (20, 21, 22, & 23 in decimal).

As for the question on using the transporters, follow these directions:

1. Enter a transporter (Press "E")
2. Enter access code ("LETSGO")
3. Enter destination code (00 thru 04)
4. Verify (Y/N)

The destination codes are as follows:

00 = Public Tracking Office
01 = Novue Apartments
02 = University Main Building
03 = Metropolis Administration
04 = Megatech Building B

If anybody has any other questions, comments or other stuff, send it in to Computist. It is the interaction that keeps my interest, and I believe that it is the best feature of the magazine? ...newsletter? ....magaletter? ...newszine? ...whatever you want to call it.

Ravi Hariprasad   NY

⚙ Mr. Supinski's softkey in issue #68 for Stickybear Math did not work for me. Everything worked up until the step 14 in which I was to scan for the bytes "60 A2 00 0A 00". When I scanned for these bytes, I got a "Bytes not Found," message from Copy II+. Anybody know what's going on?

To Chris Moffitt: In issue #68, you asked about how to make a picture BRUNable. Well the solution is quite simple! In the back of the owners manual to "Beagle BASIC" (by Beagle Brothers,) there is a short listing for a program that does just that. I would include the program here, but I am not sure about the legal implications.

Another way to get the pictures to display is to use the BLOAD, and CALL commands:

BLOAD Filename, A$2000   loads file into Hires Page 1

CALL -3100   displays page 1

For those of you beginners that don't know this already, to save a picture on page 1, you use:

BSAVE Filename, A$2000, L$1FFF
and use:

BSAVE Filename, A$4000, L$1FFF
to save a picture on Hires page 2.

And as to your other question, a couple of good machine language books for beginners (like me) are: Assembly Lines: The Book, by Roger Wagner, and Beginner's Guide to Assembly Language, by MindCraft Publishing (aka. NIBBLE).

I hope that helps. Thanks for such a great computer magazine. Keep up the good work!

I also need softkeys for WasteLand, Operation:Frog, Agent USA, and Spell It!.

B. Dudley Brett   Canada

Locksmith 6.0 Fastcopy with E.A. RWTS (Revised!)

How to Boot quickly into your binary file

To John Jackson: In COMPUTIST #56, pp.12-13, I presented a method to fastboot into Locksmith 6.0 Fastcopy. Since that time, I have enjoyed reading about updates and the difficulties in first obtaining the Locksmith Fastcopy program. I considered myself lucky, as I never had any difficulty following the original article in COMPUTIST #43. I also read with interest the discussion of my method in COMPUTIST #62, p.21, and agreed that the modification would make my program better. Then in #67, p.28, I saw a plea for help from John Jackson. As he did not specify what the problem was, I decided to try to retrace John's steps by reading my article again, and following all steps.

After incorporating the revisions of COMPUTIST #62, I faithfully followed all steps, and after doing so, I turned the computer off, placed the fastboot disk in the drive, and turned the computer on. Locksmith came up, all right, but simply would not read the source disk! Something was horribly wrong. After comparing the information on this disk with my old, well used fastboot disk, I realized that there was a typographical error (my mistake and nobody else's!). In the machine code placed on T$09, S$0F, byte $42 should be $FE, not $FF. This wee mistake means that RWTS will never be moved from $5000, where it is loaded, to $B600 - $BFFF, its proper place. Thus, the only way the disk will work is to boot DOS 3.3 first, and then use control-open apple-reset (or PR#6).

Now, after seeing this mistake, I thought again of John Jackson's problem. Was my awful error his only problem? Then I remembered the discussions about the difficulties with obtaining the Locksmith Fastcopy file. There seem to be at least 2 versions, one that starts at $2000 (my version) and another that starts at $2003. If John has the latter, he must not load it in at $2000. It should be loaded in at $2003, and the bytes $2000-$2002 should be loaded with EA EA EA (NOP bytes). Also, the file sector length should be checked by a CATALOG. Mine is 23 sectors. Who knows what length the second version is?

Here is a slightly revised (and now correct) version of the cookbook method:

The Annotated Procedure

1. Format a DOS 3.3 disk (Use Locksmith to format, it's quick.)

2. Copy track $00 of any Electronic Arts disk to the formatted disk (I used Seven Cities of Gold).

3. Get out a sector editor (CIA?) and edit track $00, sector $00.

Trk	Sct	Byte(s)	From		To	
00	00	04-12	A9 00 8D F2		2C E9 C0 A9	
			03 A9 C6 8D		18 85 3E A9	
			F3 03 49 A5		09 20 00 0C	
			8D F4 03		20 00 18	

This translates in disassembly to:

2C E9 C0	BIT $C0E9	(Turn on drive)	
A9 18	LDA #$18	(Page no. for special loader)	
85 3E	STA $3E	(Store $18 in loc. $3E)	
A9 09	LDA #$09	(Track no. for special loader)	
20 00 0C	JSR $0C00	(E.A. Load from track $09)	
20 00 18	JSR $1800	(Execute loader at $1800)	

4. Now edit track $09, sector $0F starting at byte $00 (This is the special loader)

Trk	Sct	Byte(s)	From	To
09	0F	00	??	A9 20 85 3E A9
				20 20 00 0C A9
				30 85 3E A9 21
				20 00 0C A9 50
				85 3E A9 22 20
				00 0C 2C E8 C0
				A9 00 85 3C A9
				50 85 3D A9 FF
				85 3E A9 59 85
				3F A9 00 85 42
				A9 B6 85 43 20
				4A FF A9 00 85
				47 20 3F FF 20
				2C FE 4C 00 20

Disassembly listing

```
xx00: A9 20    LDA #$20    (Address of LS 6.0
                            Fastcopy)
02: 85 3E      STA $3E     (Store $20 in loc. $3E)
04: A9 20      LDA #$20    (Track no. for LS)
06: 20 00 0C   JSR $0C00   (Read in track $20)
09: A9 30      LDA #$30    (2nd half of Locksmith)
0B: 85 3E      STA $3E
0D: A9 21      LDA #$21    (Track no. for 2nd half)
0F: 20 00 0C   JSR $0C00
12: A9 50      LDA #$50    (Address of DOS 3.3
                            RWTS)
14: 85 3E      STA $3E
16: A9 22      LDA #$22    (Track with RWTS)
18: 20 00 0C   JSR $0C00
1B: 2C E8 C0   BIT $C0E8   (Turn off drive)
1E: A9 00      LDA #$00    (Low 1st byte of RWTS)
20: 85 3C      STA $3C
22: A9 50      LDA #$50    (High 1st byte of RWTS)
24: 85 3D      STA $3D
26: A9 FF      LDA #$FF    (Low last byte of RWTS)
28: 85 3E      STA $3E
2A: A9 59      LDA #$59    (High last byte of RWTS)
2C: 85 3F      STA $3F
2E: A9 00      LDA #$00    (Low byte of destination)
30: 85 42      STA $42
32: A9 B6      LDA #$B6    (High byte of destination)
34: 85 43      STA $43
36: 20 4A FF   JSR $FF4A   (Start memory move)
39: A9 00      LDA #$00
3B: 85 47      STA $47
3D: 20 3F FF   JSR $FF3F
40: 20 2C FE   JSR $FE2C   (End memory move)
                           (byte $42 change)
43: 4C 00 20   JMP $2000   (Execute LS 6.0
                            Fastcopy)
```

My original plan was to load Locksmith into tracks $20 and $21. After doing so I found that Locksmith needed the DOS 3.3 RWTS from $B600 to $BFFF. Thus, I eventually decided to place RWTS on track $22 and have the special loader retrieve it from that track and place it in memory starting at $5000. Then the loader would make a memory move, transferring RWTS to $B600.

5. On another DOS 3.3 disk enter and save (SAVE E.A.WRITE) the following:

WRITE BINARY FILE

```
10 REM Write Hi-Res picture or binary
   file
20 POKE 768,32: POKE 769,227: POKE
   770,3: POKE 771,76: POKE 772,217:
   POKE 773,3: POKE 47083,0: POKE
   47088,0: POKE 47090,0: POKE
   47091,0: POKE 47090,2
30 HOME : INPUT "TRACK~#:~" ;X: INPUT
   "#~OF~SECTORS:~" ;Y: PRINT : PRINT
   : PRINT "INSERT~DISK..." ; : GET
   PP$: PRINT
40 T = X:TF = Y:SI = 15:MI = 32:MF =
   MI + Y + 1:S = SI:M = MI:LT = T +
   INT (Y / 16): GOSUB 80
50 S = S - 1: IF S < 0 THEN S = SI:T
   = T + 1: IF T > LT THEN PRINT : - ERR
   OR - "":END
60 M = M + 1: IF M > MF THEN END
70 GOSUB 80: GOTO 50
80 POKE 47089,M: POKE 47084,T: POKE
   47085,S: CALL 768: RETURN
```

Checksums

10-$BADD	40-$D4C8	70-$7D67
20-$2B54	50-$6B1C	80-$B2BA
30-$8AA8	60-$4A87	

6. Now boot DOS 3.3 and BLOAD LOCKSMITH 6.0 FASTCOPY

CATALOG (Check sector length of file)
BLOAD LOCKSMITH 6.0 FASTCOPY
CALL-151
AA72.AA73 (Location of starting address)

If you see "AA72:00 20" then skip the following line and go directly to step 6. If you see "AA72:03 20" then do the following.
2000:EA EA EA

7. Now write Locksmith to your prepared E.A. boot disk.
LOAD E.A.WRITE   Never run it on the disk you saved it!

Put your prepared E.A. boot disk into the drive and:
RUN

Answer 32 when prompted for the track#. Answer 23 when prompted for sectors (check this!).

8. Now write RWTS to the disk.
CALL-151
2000<B600.BFFFM   Move RWTS into place for transfer
3D0G
RUN   **Make sure E.A.WRITE is in place**

Answer 34 when prompted for the track. Answer 10 when prompted for sectors.

9. Now place the new boot disk in the drive, boot and enjoy!

A Bug in Magic Spells softkey COMPUTIST #66, p.35-37

When I found the one byte error in the Fastboot article, I decided to double check any other articles in which I had used the monitor move routine. The only other article with this routine was a rather long article in COMPUTIST, v.66, about the torture I exposed myself to in softkeying Magic Spells & Moptown of Special Delivery Software. There in the listing and disassembly of the MVE file was the same one byte typo error! Somehow, in the writing of the article, I had consulted old notes, instead of taking the exact listing from the MVE file.

Here is the correction: In the hexdump (p.36) and source code listing (p.37) of the binary file "MVE", change byte $0324 from $FF to $FE.

By the way, I have noted in COMPUTIST some recent softkeys for Magic Spells distributed by Learning Company. I do not know if these refer to an updated version of the very old version (1981) that I softkeyed. It is obvious, however, that the recent protection methods are much simpler than in my version.

Softkey for...

Crosscountry Canada

Didatech Software Ltd.

Crosscountry Canada is an interesting Geography program introducing students to places and their products in Canada. Both sides may be copied with a sector copy program. One sector (T$02 - S$07), however, produces errors and the disk will not boot unless this sector is copied. It can be done by copying track $02 with a Bit Copy. As such copies seem to fail easily, I finally decided to try and remove all protection.

After finding that I could not read the protected sector (using C.I.A.), I searched the disk's DOS to find the HELLO file (shown on T$01, S$07) and found it to be a binary file called COPYRIGHT 1986. As there was a normal catalog, I then BLOADed it and tried to read it. To my surprise, excepting the first 3 bytes, the file was garbage. These first 3 bytes (0802:4C E1 BC) simply caused a jump to a normally unused portion of DOS 3.3. I decided that perhaps a decoding routine might be at that location, effectively to normalize the hello file. At this point, using a modified F8 ROM, I booted the Crosscountry Canada original disk, reset into the monitor, and examined the code at $BCE1. All I could see were zeros!

By examining the code prior to these unhelpful zero bytes, it was apparent that this was almost the normal DOS 3.3. Perhaps I could get a better look by using a sector editor. Using C.I.A. again, I found page $BC on T$00, S$06, and there it was, from $BCE1 to $BCFB, a nice decoding routine that substituted bytes EOR'd with $96 in the hello file. At the end of this routine a jump was accomplished ($BCF9:4C 89 BA) to location $BA89, again in DOS.

Using the sector editor, I found page $BA on T$00, S$04, and looking at the routine from $BA89 to $BA94, another twist was found. This tiny routine simply zeros the 27 bytes of the decoding routine at $BCE1 and exits by jumping to location $0805 of the hello file ($BA92:4C 05 08).

Deciding that I could be just as sneaky, I placed a zero byte at $BA92 on a copy of the disk. Then I booted this copy and when it crashed into the monitor, I simply replaced the original encoded file on the disk with the now decoded file by bsaving it on the disk. I purposely saved it starting at $0805, instead of $0802, as I didn't need the decoded routine anymore.

Now I could examine the nibble count that undoubtedly was in the hello file. After bloading it, I started looking through the monitor listing for disk access, and found the nibble count from location $0A70 to $0B44. Noting frequent branches in this subroutine to $0B33, I discovered at this location that the Y-register is loaded with $FE and if the nibble count is successful, the contents of the register are placed at $1FFA and an RTS is made. However, upon an error, the Y-register is incremented (to $FF) before the return.

Having found the nibble count, I then scanned back through the file to find where the nibble subroutine was accessed. At location $088E, there it was ($088E:20 70 0A JSR $0A70). Immediately after this, the Y-register is incremented and checked to see if it is $FF. If not, a branch is made to disk death at $0888. If successful, the disk is now unlocked!

My method of deprotection was now clear: Edit the nibble count so it would return $FE always and edit the branch to disk death using NOP bytes. Here then is the cookbook recipe to completely deprotect Crosscountry Canada.

1. Copy both sides of the original with any fastcopier that ignores errors on T$02, S$07.

2. Sector edit T$00, S$04, byte $92 from $4C to $00 (This prevents the decoded file from being executed after decoding).

3. Boot the now altered side one copy, and wait for the monitor asterisk prompt. Then edit the hello file in memory.

```
0899:EA EA      was D0 ED, removes branch
0B37:EA         was C8, removes the INY
UNLOCK COPYRIGHT 1986
BSAVE COPYRIGHT 1986, A$0805, L$373
   Save the modified and decoded hello file
LOCK COPYRIGHT 1986
```

DOS 3.2 Controller

Bill Jetzer (in his article "Reading from Protected Disks", COMPUTIST #60 p.11) included a controller to convert DOS 3.2 disks to DOS 3.3. I tried to use this controller with Super IOB v1.5 to access the files on on old 3.2 disk (Game Show) and it would not work. I examined the operations ordered by the controller and found that two GOSUBs referenced the wrong entry points in SIOB1.5. GOSUB 430 in lines 1012, 1014, 1020 and 1030 will not exchange RWTS routines as suggested and should be changed in all cases to GOSUB 360. GOSUB 460 in lines 1020 and 1030 simply prints a message, waits for a keypress and returns. GOSUB 460 must be changed to GOSUB 490 in both places to correctly toggle the read/write pointer. Additionally, when I tried this controller on Game Show, I found that it would read track $00 fine, but would bomb on track $01. Noting that the bomb occurred as Super IOB was trying to read sectors $0F to $0D, I researched the documentation on the copy program. I discovered in the binary file, IOB.OBJ0, that after a track was read or written, the sector number was reset to $0F (15). This is necessary for DOS 3.3, but for DOS 3.2, it should be $0C (12)! Thus a one byte patch must be made. $0F should be changed to $0C at address $384. Note that this is accomplished in the controller in line 1011.

Here is the modified DOS 3.2 Controller, CON.13. Note that line 1012 includes 4 POKEs that I used to copy Game Show by Computer Advanced Ideas. These turn off the error checks for the first bytes of the address and data markers, the read error flag and data checksum. I was successful, by the way, in accessing all files of Game Show, but as yet have still not made a runable version, converted to DOS 3.3. However, from the extensive research I had to do, I finally composed Parms for Copy II Plus that would faithfully and reasonably quickly make a protected copy.

Controller

```
1000 REM DOS3.2 CONTROLLER
1010 TK = 0:LT = 35:ST = 12:LS = 12:CD
     = WR:FAST = 1
1011 GOSUB 360: REM SWAP RWTS ROUTINES
     TO MAKE MODIFICATIONS
1012 POKE 900,12: REM 13 SECTOR PATCH
     IN IOB.OBJ0
1013 POKE 47480,0: POKE 47370,0: POKE
     47459,24: POKE 47439,0: REM DOS3.2
     RWTS PATCHES
1014 GOSUB 360: REM SWAP RWTS ROUTINES
     AGAIN
1020 GOSUB 360: GOSUB 490: GOSUB 610
1030 GOSUB 360: GOSUB 490: GOSUB 610:
     IF PEEK (TRK) = LT THEN 1050
1040 TK = PEEK (TRK):ST = PEEK (SCT):
     GOTO 1020
1050 HOME : PRINT "COPY~DONE" : END
10010 PRINT CHR$ (4)
     "BLOAD~RWTS.13,A$1900"
```

Checksums

```
1000-$356B  1013-$C8FE  1040-$83C2
1010-$135A  1014-$917B  1050-$6B2F
1011-$184E  1020-$7C36  10010-$EF46
1012-$13F5  1030-$4EA2
```

Bitkey for...

Game Show

Advanced Ideas Inc.

The Game Show is a delightful quiz game based upon numerous TV game shows. It features a moderator and one or two panelists and an extensive selection of questions in many categories. For a 1982 program, the graphics are good.

Bit copies of this program are produced very slowly and are very unreliable. Most bit copiers will not even copy it. For this reason, I have devoted a great deal of effort whenever I could obtain time over a time of several months. There are several layers of protection which have made my attempts frustrating. Some of the more important protection methods found are:

1. The disks are all in DOS 3.2 format.

2. The program disk has altered prologues: I found address and data prologues of FE AA B5 and FE AA AD, respectively, on tracks $01-$10, $12-$13 and D5 AA B5 and D5 AA AD on tracks $11 and $14-$22. Track $00 was a nightmare. Sector $00 is DOS 3.3 having normal prologues. However, a gibberish sector appears if DOS 3.3 marks are used. Sectors $01-$09 have normal

prologues (D5 AA B5 and D5 AA AD), whereas sectors $0A-$0C are changed to FE AA B5 and FE AA AD.

3. At least 2 sectors on the program disk have data checksum and data epilogue errors.

4. There are two catalogues, one on Track $11, the other on Track $08. A routine starting at $A800 in the disk's DOS is called to switch prologue bytes and track numbers in order to access both catalogues.

5. Machine language routines, mainly for graphics, are loaded into $800-$FFFF and $1800-$1FFF when the disk is booted and are not shown in the catalog.

6. Instead of using Applesoft programs, this disk uses Integer Basic. This is really a problem, as Integer Basic is not manipulated in RAM as an Applesoft program might. Instead of running low in memory at $801, Integer runs high in memory. Variable manipulation is also quite different.

Considering the above, I was able to produce a DOS 3.3 copy of GAME SHOW that, of course, would not boot, using CON.13, the included controller. Note that a special RWTS, RWTS.13, must be used in this swap controller. In order to obtain this, I used Computist's method from v.13, p.27. I then used FID to transfer files from the copied disk's T$11 to an initialized DOS 3.3 disk. In order to copy the Integer files on T$08, I booted up an old DOS 3.3 master to obtain the Integer Basic prompt, and changed the catalog track with a POKE 44033,8. In turn, I loaded the files from the copied disk and after changing back to the normal track (POKE 44033,11), I saved them on to the disk with the FIDed files. At this point, I even went so far as to transfer the necessary Integer files from the system master, and made a HELLO file to provide Integer Basic on this disk.

Inspection of the Integer files, and the realization that several machine language routines not in the catalog were being accessed, convinced me that I would never, not in a hundred years, be able to convert this program into a runable DOS 3.3 environment!

I now was almost stymied until I realized that my knowledge of the marker changes could be used to provide a PARM for Copy II Plus that would provide a good copy, though it would still retain its protection. With this in mind, I planned a PARM that would sector copy all tracks but T$00. As T$00 had a combined DOS 3.2/3.3 format and extensive prologue changes, I decided that it should be bit copied. PARM GS.1 can be entered into any recent version of Copy II Plus and will faithfully copy the program side. PARM GS.2 will copy any data diskette.

PARM GS.1

GAME SHOW * (CAI)
```
T0
T11, SECTOR COPY, 59=B5, 6B=00, 65=00,
     77=00
T14-T22, SECTOR COPY
T1-T10, SECTOR COPY
T12-T13, SECTOR COPY
```

PARM GS.2

GAME SHOW-DATA * (CAI)
```
T0-T22, SECTOR COPY, 59=B5, 6B=00, 65=00,
     77=00
```

Softkey for...

Computer Inspector v1.0

MECC 1988

This MECC disk has the data prologue altered from D5 AA AD to D5 AD AA. I used COPYA and a sector editor to produce a copyable backup.

1. Run COPYA and then stop it to make some mods.
```
RUN COPYA
Ctrl C
70
CALL -151
B8F3:00      turn off check for byte 2 of data pro-
             logue
B8FE:00      turn off check for byte 3 of data pro-
             logue
ctrl C
RUN
```

2. Sector edit:

Trk	Sct	Byte(s)	From	To
$00	$0B	99	AD	AA
		A3	AA	AD
	$0C	82	AD	AA
		87	AA	AD

Enjoy!

Softkey for...

Comment Dit-On?

Lingo Fun, Inc.

After first checking this disk out for changed address and data markers, I found that track $03 was sectorless and track $11, the catalogue track, had epilogue bytes changed to D5 AA from DE AA. Accessing the catalog, I was interested to find 2 files that triggered my memory. These files were SSPROT$$A and SSPROT$$1, used in protection many times before. Researching old COMPUTIST editions, I noted the resemblances between the protection on Comment Dit-On and on Walt Disney's Cartoon Maker, softkeyed by Edward Teach in v.51, p.28. Though some minor details were different, the technique of protection

shown was remarkably similar. I found that:
- track $03 is unreadable and likely a nibble count track.
- track $11 has minimally altered epilogues
- Byte 01 of T$11 S$00 is 00 and should be 11 to access the catalogue.
- The HELLO file BRUNs SSPROT$$1 to check before proceeding.
- Another file (the real HELLO program) is not mentioned in the catalogue, but can be found with patience.

On Track $12, sector $0F, I discovered the T/S list of the real HELLO file that runs after SSPROT$$1 has done its dirty work. I decided to simply exchange this 4 sector file in the catalogue for the 2 sector file that calls the protection. Here is the method.

1. INIT both sides of a disk with HELLO, deleting the file afterwards.

2. Copy both sides of the original disk with a fast copy program that will ignore track $03 errors. As we need the VTOC and catalogue on track $11, make sure to change DOS 3.3 first by poking *B942:18 to ignore the epilogue errors on track $11.

3. Sector edit both sides of the copied disk:

Side one:

Trk	Sct	Byte(s)	From	To	
$11	$00	01	00	11	
	0B	DD	21	12	(swap HELLOs)
		FE	02	04	

Side two:

Trk	Sct	Byte(s)	From	To
$11	00	01	00	11
	0E	97	1B	12
		B8	02	04

4. Boot DOS 3.3 and turn off error check in DOS.
```
POKE 47426,24
BRUN FID
```

5. Transfer all files from the altered copied disk to both sides of your initialized disk.

Softkey for...

2400 AD

Origin

I have noted that the type of protection on this disk has received considerable attention and would like to express the idea that the elaborate approaches to deprotection are somewhat of an overkill. Specifically, when address prologues are alternated, D5 AA 96 for even tracks and D4 AA 96 for odd tracks, all that needs to be done is to turn off the check for the first address byte in DOS 3.3 and use COPYA.

1. Start COPYA.
```
RUN COPYA
ctrl C
70
CALL-151
B957:00   turn off check for 1st address prologue
          byte
ctrl C                         back to basic
RUN                            copy both sides
```

Softkey for...

Magic Slate v1.3

Sunburst

In COMPUTIST v.49, p.26, I commented upon v1.2.3 and noted the moved, but almost identical to the original, protection. Again in v1.3, the routine has been moved. Here is the updated copy method for the 20 and 40 column programs.

1. Boot PRODOS
```
BLOAD MS,TSYS,A$2000
CALL-151
2053:EA EA EA          was 4C 2F 21
205B:EA EA EA          was 4C 2F 21
BSAVE MS, TSYS, A$2000, L$3DEC
```

2. Repeat step 1 for other side.

Softkey for...

French: Verb Pairs and Idioms

Language Study Software

The data epilogues are changed from DE AA to DE DE in the minimal protection observed.

1. Boot a DOS 3.3 disk. Place a blank disk in the drive and initialize a slave disk.
```
INIT HELLO
DELETE HELLO
```

2. Get rid of some error checking and use FID to transfer all files to the initialized disk.
```
POKE 47426,24
BRUN FID
```

Softkey for...

Sports Stats

MECC,

Having deprotected many MECC disks in the past, I was expecting to spend perhaps 5 minutes to manage Sports Stats, a 1985 program. At first, the only protection visible was changed address prologues (AA D5 96 instead of D5 AA 96) and changed data prologues (D5 AD AA instead of D5 AA AD). So, following usual procedure, I booted the disk, waited until the Applesoft prompt appeared and hit <RESET> several times. I moved

the RWTS down (*1900<B800.BFFFM) and continued by installing the MECC swap controller and using SUPER IOB to produce a COPYAable disk. After scanning Track $00, I edited a few bytes so that only legitimate prologue bytes would appear in the MECC DOS3.3 and then booted my copy.

Darn! The disk booted, but crashed after the menu appeared. Something was different in the copy protection! I also noted that the MECC logo was quite different from all others that I had seen before. After considerable further investigation, I found that the program used a CALL (CALL 30464,1 (or 2)) to a binary file, MECC FLIPPER, to switch prologue bytes so that both the original protected disk and an unprotected data disk could be accessed. To further complicate matters, each time a disk was read, it was checked for whether it was a protected, or an unprotected disk. Simply altering the MECC FLIPPER program to not exchange bytes was insufficient; I had to modify the BASIC programs to ensure that the disk checks were made, but without the checks for whether the disk was protected or not. Here then is the method:

1. Boot the original disk and hit <RESET> several times to stop the drive.
```
CALL-151
1900<B800.BFFFM      move the RWTS down
```

2. Boot a DOS 3.3 disk without a HELLO program
```
BSAVE SPORTS.RWTS, A$1900, L$800 Save
     RWTS
```

3. Install the MECC Swap Controller into Super SIOB and copy the original to a blank disk.

4. Make the following sector edits:

Trk	Sct	Byte(s)	From	To
$00	$02	$58	AD	AA
		$5F	AA	AD
		$F1	AD	AA
		$FC	AD	AA
$00	$03	$55	AA	D5
		$5F	D5	AA
$00	$06	$7A	AA	D5
		$7F	D5	AA

5. Boot DOS 3.3 and insert copied disk.
```
BLOAD MECC FLIPPER
CALL-151
7709:60      was A5, aborts prologue switches
ctrl C
UNLOCK MECC FLIPPER
BSAVE MECC FLIPPER, A$7700, L$CD
LOCK MECC FLIPPER

LOAD HELLO-OPTIONS
UNLOCK HELLO-OPTIONS
2110 ONERR GOTO 2130
2120 GOSUB 500:PRINT D$"VERIFY
     DELETIONS"
2130 IF PEEK (222) = 6 THEN PD = 0:
     GOTO 3110
4610 ONERR GOTO 4625
4620 PRINT D$"VERIFY DELETIONS,D"DR:
     GOTO4670
4625 IF PEEK (222) = 6 THEN PD =0: GOTO
     5000
SAVE HELLO-OPTIONS
LOCK HELLO-OPTIONS

LOAD STIX V52
UNLOCK STIX V52
3508 ONERR GOTO 3512
3511 PRINT C$"VERIFY DELETIONS,D"DR:
     PD = 1: GOTO 3544
3512 IF PEEK (222) = 6 THEN PD = 0:
     GOTO 3544
7271 ONERR GOTO 7274
7273 PRINT CHR$(4)"VERIFY
     DELETIONS,D"DR: GOTO 7283
7274 IF PEEK (222) = 6 THEN PD = 1:
     GOTO 7295
SAVE STIX V52
LOCK STIX V52
```

The Mandelbrot Set

No, this is not another softkey; there is no commercial or private domain program called The Mandelbrot Set, to the best of my knowledge. Rather, I wish to take you on a journey into the realm of higher mathematics, non-linear equations and the science of Chaos, the study of the unpredictable.

First, I should tell you how it all began. During March Break this year (I am a high school Physics teacher), I visited a bookstore to buy some books to while away the vacation period. There I came upon a book on sale, with a most unusual title. It was called "CHAOS, Making a New Science" by James Gleick, an editor and reporter at The New York Times. Sensing a bargain, I bought it, and during the Break, I read it and was amazed at the new (to me, at least) approaches to solving problems that heretofore seemed not to be predictable and therefore not solvable (An example might be the dynamics of turbulent stream flow). The book is well written and generally non-mathematical and can be read with pleasure by most curious persons of reasonable intelligence.

Part way through the book is a section that might interest the readers of COMPUTIST. There the work of Benoit Mandelbrot upon the Fractal Geometry of Nature is presented. In 1977, Mandelbrot presented the scientific world with a new approach to the study of some of the convoluted patterns that appear in nature (examples might be the endlessly convoluted outline of a fern frond or the patterns of snowflakes, no two alike!). One pattern produced by Mandelbrot, called the

Mandelbrot Set, is so complex that its appearance on a computer Monitor screen produces awe. The neat thing is that, if any part of the perimeter of this figure is magnified, no matter how much, there is always incredible detail. The perimeter, then is so convoluted that one can only infer that it is infinitely long!

In order to present a graphic image of the Mandelbrot Set on a computer monitor, at least 10 million calculations must be performed. As no one in control of their wits would want to do these computations, the computer is thus the tool to complete the job. Even so, it takes several hours for a BASIC program on an APPLE to do it. Compiling the program reduces the wait to 1 1/2 hours (the Macintosh takes 13 minutes!). Resolution of the finished image on the Hires screen of the Apple II is not nearly as spectacular as when one tries an IBM or Amiga, but is not bad.

To generate this image, one can simply "RUN MANDELBROT.1" or come with me on a journey through mathland as I explain how to calculate it. If you are also interested in seeing a blowup (80X) of the little dot in the far left part of the whole image, then "RUN MANDELBROT.2". For those who have a color monitor, "MANDELBROT.3" presents the image with a multicolored border.

Here is the mathematics of the Mandelbrot Set: The Mandelbrot Set is an image which falls in the range of coordinates X values from -1.9 to +0.5 and Y values from -1.10 to +1.10. Think of the X value as representing a direction on a map. If X is negative, this is towards the west; if X is positive it is directed towards the east. A positive Y value, then is towards the north (or up the monitor screen) and a negative Y value is to the south. In the computer program, one simply tests every coordinate point in the range given in order to see if the point tested lies inside, or outside of the Mandelbrot Set. If the point lies inside, it is plotted; if it lies outside, it will not be plotted.

When one wishes to test a point, say X=0.25 and y=-.53, one simply sets a variable, A to the X value and another variable, B to the Y value. The weird thing, however, is that whereas A is a Real number, B is something mathematicians call an imaginary number. Fear not, for I will explain without confusing anybody.

The way to see how these two types of numbers, and their use is to show a simple statement. The coordinates of each tested point can be represented this way:

Coordinates of point = Real no. + Imaginary no.
C = A + B

Using our above example the tested point's formula is:

C = 0.25 + -0.53

So to test each point, simply set up the easy formula, with the coordinates placed in their correct place. Now to test the point to see if it fits the Set, take this expression (A + B) and square it, and then add the original expression to the product.

Now if one were dealing with only real numbers, one would get the following: (the product part)

(A + B) x (A + B) = A^2 + 2.A.B + B^2

and now adding the original expression:

(A^2 + 2.A.B + B^2) + A + B = A^2 + 2.A.B + B^2 + A + B

Note if our original seed values were real numbers, we would get:

.25^2 + (2 x .25 x -.53) + -.53^2 + .25 + -(.53) =
.0625 + -.265 + .2809 + .25 - .53 = -.2016

But since we are using 1 real number and 1 imaginary number, we have to do something a bit different. You have to know the simple rules of dealing with them. These are the rules:

1) Add a real number to another real one gives a real number

2) imaginary + imaginary number gives an imaginary number

3) real times real number gives a real number

4) imaginary times imaginary gives a real negative number!!!

This last rule is the nice one; it provides a link between real and imaginary numbers. Let's use it now in our original example. To make things easier to follow, however, let us put a flag against any number which is imaginary. Thus, the original example of an imaginary number (-0.53) is shown as -.53i. The little "i" then is the identifying flag. Now let's see the coordinate formula:

coord. = A + B

giving us:
= 0.25 + -0.53i

Now square it
(.25 + -.53i) x (.25 + -.53i) =
(.25 x .25) + (.25 x -.53i)x2 + (-.53i x -.53i) =
.0625 + -.265i + -.2809 = -.2184 - .265i
Note by rule 4 that
(-.53i)^2 = -.2809 *(1i x 1i = -1)

Now add the original A + B expression:
(-.2184 - .265i) + (.25 + -.53i) = .0316 + -.795i

To continue the test, you must take this new expression (.0316 - .795i) and square it again and add the ORIGINAL expression to it:

(.0316 - .795i) x (.0316 - .795i) + (.25 - .53i)

Do this over and over again, at least twenty times. If during these calculations, the real number or the imaginary number rises above 2 (positive or negative 2), this is a sign that the number will

eventually become infinitely larger, and must therefore lie outside the Mandelbrot Set. The number is therefore discarded and not plotted. But if the numbers never attain a value greater than 2, the coordinate point lies inside the Set and is plotted.

In the program, MANDELBROT.1, lines 70 to 100 set up the Hires screen and adjust the coordinate points to equation and screen values. Line 110 sets the loop to do 20 repeat calculations testing the point. Lines 120 to 140 perform one calculation and line 150 tests the point for size and discards the point if too large by exiting from the loop. If the point passes the test, HCOLOR is set to 3 and is plotted in lines 160 and 170. If the point does not pass, then HCOLOR is set to 0 and plotted (invisibly). Line 180 loops back to start testing another coordinate.

MANDELBROT.1

```
10 REM ****************
20 REM * MANDELBROT.1 *
30 REM * BY B.D.BRETT *
40 REM * MAY 18, 1989 *
50 REM ****************
60 TEXT : HOME
70 HGR2 :X1 = 180:Y1 = 86
80 FOR X = - 160 TO 40
90 FOR Y = 0 TO 85
100 S = 80:A = X / S:B = Y / S:D = A:E
    = B
110 FOR I = 1 TO 20
120 C = A
130 A = A * A - B * B + D
140 B = 2 * C * B + E
150 HCOLOR= (I > 19) * 3:I = I + ((
    ABS (A) > 2) + ( ABS (B) > 2)) * 20
160 HPLOT X + X1,Y1 - Y
170 HPLOT X + X1,Y1 + Y
180 NEXT I,Y,X
190 PRINT CHR$ (7); CHR$ (7): END
```

Checksums

10-$BADD	80-$A1E5	150-$9763	
20-$9B13	90-$862B	160-$88F5	
30-$4D3B	100-$9BFE	170-$488A	
40-$AD92	110-$3996	180-$23DA	
50-$C899	120-$00A6	190-$E044	
60-$DA01	130-$298A		
70-$1D72	140-$6A7F		

MANDELBROT.2 shows a magnified view of an obscure "bump" on the east-west line near the left side of the screen. Here I simply set the range over a much smaller range of X & Y values. Note the fine exotic detail persists in this view. Interested programmers may extend the technique to obtain increasingly magnified views that will always show feathery strange detail.

MANDELBROT.2

```
10 REM ****************
20 REM * MANDELBROT.2 *
30 REM * BY B.D.BRETT *
40 REM * MAY 18, 1989 *
50 REM ****************
60 TEXT : HOME
70 HGR2 :S = 3700:T = 6690:U = 90:F =
    180
80 FOR X = 10 TO 250
90 X1 = X - T
100 FOR Y = - 90 TO 0
110 Y1 = U + Y:A = X1 / S:B = Y1 / S:D
    = A:E = B
120 FOR I = 1 TO 20
130 C = A
140 A = A * A - B * B + D
150 B = 2 * C * B + E
160 HCOLOR= (I > 19) * 3:I = I + ((
    ABS (A > 2) + ( ABS (B) > 2)) * 20
170 HPLOT X,Y - Y: HPLOT X,Y + F
180 NEXT I,Y,X
190 PRINT CHR$ (7); CHR$ (7): END
```

Checksums

10-$BADD	80-$758B	150-$6E3F	
20-$9B13	90-$1BEC	160-$A4B2	
30-$4D3B	100-$D769	170-$51B3	
40-$AD92	110-$653C	180-$6AB3	
50-$C899	120-$1173	190-$7B1B	
60-$DA01	130-$2BDF		
70-$9BD2	140-$ECDE		

MANDELBROT.3 is a colored variation of MANDELBROT.1. Here, I have plotted points in different colors, inside and outside the Set. I accomplished this by plotting points that failed the test in only one calculation in green, the points failing after 3 calculations in purple, etc., and not plotting the points which passed the test even after 20 calculations.

MANDELBROT.3

```
10 REM ****************
20 REM * MANDELBROT.3 *
30 REM * BY B.D.BRETT *
40 REM * MAY 18, 1989 *
50 REM ****************
60 TEXT : HOME
70 HGR2 :X1 = 180:Y1 = 86
80 FOR X = - 160 TO 70
90 FOR Y = 0 TO 85
100 S = 80:A = X / S:B = Y / S:D = A:E
    = B
110 FOR I = 1 TO 20
120 C = A
130 A = A * A - B * B + D
140 B = 2 * C * B + E
150 HCOLOR= (I > 0) + (I > 3) + (I >
    6) + (I > 9) * 2 + (I > 12) + (I >
    14) - (I = 20) * 3
160 HPLOT X + X1,Y1 - Y: HPLOT X +
    X1,Y1 + Y
170 I = I + (( ABS (A) > 2) + ( ABS
    (B) > 2)) * 20
180 NEXT I,Y,X
```

```
190 PRINT CHR$ (7); CHR$ (7): END
```

Checksums

10-$BADD	80-$A6E6	150-$DE2D
20-$9B13	90-$9B1C	160-$3E0D
30-$4D3B	100-$ACE3	170-$2B46
40-$AD92	110-$3A91	180-$B056
50-$C899	120-$0DA1	190-$6BDE
60-$DA01	130-$288B	
70-$1D72	140-$7748	

I compiled these programs, using the Beagle Compiler and ProDOS, and decreased the run-time to about 1 1/2 hours. If uncompiled, the computer should be left on overnight. The same program can be easily adapted to a higher resolution computer (I tried it on a MacIntosh and obtained a beautiful result).

For programmers who have taken interest in one-line programs as published in NIBBLE, I have compressed each of the three programs (See MB1.1 LINER, MB2.1 LINER & MB3.1 LINER). I decided to compose these after looking through an old NIBBLE magazine (Aug.,1988) a few days after starting to write this article. There, on page 110, was a one-liner that drew a Mandelbrot Set! My MB3.1 LINER has close similarities to Theodore Yapo's routine, but shows much finer detail than his. An additional benefit of using such one-liners is that they run much, much faster than many line programs. When compiled, you can see a finished product in less than an hour.

Mandelbrot.1

```
1 TEXT : HOME : HGR2 :X1 = 180:Y1 =
  86: FOR X = - 160 TO 40: FOR Y = 0
  TO 85:S = 80:A = X / S:D = A:B = Y
  / S:E = B: FOR I = 1 TO 20:C = A:A
  = A * A - B * B + D:B = 2 * C * B +
  E: HCOLOR= (I > 19) * 3:I = I + ((
  ABS (A) > 2) + ( ABS (B) > 2)) *
  20: HPLOT X + X1,Y1 - Y: HPLOT X +
  X1,Y1 + Y: NEXT I,Y,X
```

Checksums

```
1-$4F2D
```

Mandelbrot.2

```
1 HGR2 :S = 3700:T = 6690:U = 90:F =
  180: FOR X = 10 TO 250:X1 = X - T:
  FOR Y = - 90 TO 0:Y1 = U + Y:A = X1
  / S:D = A:B = Y1 / S:E = B: FOR I =
  1 TO 20:C = A:A = A * A - B * B +
  D:B = 2 * C * B + E: HCOLOR= (I >
  19) * 3:I = I + (( ABS (A) > 2) + (
  ABS (B) > 2)) * 20: HPLOT X,Y: -
  HPLOT X,Y + F: NEXT I,Y,X
```

Checksums

```
1-$98DE
```

Mandelbrot.3

```
1 HGR2 :X1 = 180:Y1 = 86: FOR X = -
  160 TO 70: FOR Y = 0 TO 85:S = 80:A
  = X / S:B = Y / S:D = A:E = B: FOR
  I = 1 TO 20:C = A:A = A * A - B * B
  + D:B = 2 * C * B + E: HCOLOR= (I >
  0) + (I > 3) + (I > 6) + (I > 9) *
  2 + (I > 12) + (I > 14) - (I = 20)
  * 3: HPLOT X + X1,Y1 - Y: HPLOT
  X1,Y1 + Y:I = I + (( ABS (A) > 2) +
  ( ABS (B) > 2)) * 20: NEXT I,Y,X
```

Checksums

```
1-$F0F0
```

Deprotecting MicroLab Disks
(Micro Fun)

For several months, COMPUTIST had a number of MicroLab programs on the most wanted list. I had deprotected four of these about two years ago, but neglected to take notes. As I remembered that they were extensively protected, I have only now found enough time to take another look at them. Here are my results.

Softkey for...

Ming's Challenge
Micro Fun

Ming's Challenge has a protection somewhat similar to that used for Death in the Caribbean (COMPUTIST v.23 p.9) and Miner 2049'er (COMPUTIST v.22 p.10). Track $01 is normal, whereas T$01-T$22 have address prologues altered to D3 96 F2, data prologues to 96 D3 E5 and address and data epilogues respectively to D3 B2 and 96 FF. There is a nibble count on T$01, S$09 very much like that of Miner 2049'er, but in a different sector.

Two major differences are shown from previous protection schemes of MicroLab. There is a 2 byte alteration of the translate tables that must be fixed to allow one to read the protected tracks. Another unusual trick found is a minor change in the way marker nibbles are compared to a value in DOS. Instead of directly comparing a nibble (C9 D3 CMP #$D3), in several cases the routine compares with a value placed in page $00 at $31 (C5 31). The value held in $31 is changed from one value to another a few times. As this is done outside the RWTS, one cannot use a swap controller to copy the disk. However, a modified controller that will read using the protected disk's prologues and epilogues, as well as the altered translate bytes, will do the job. Several sector edits must also be made. Here are the steps needed to softkey Ming's Challenge:

1. Install the controller, CON.MING listed here into Super IOB and run the resulting program.

When asked whether to format the disk, answer "Y" for yes.

2. The controller will complete sector edits in track $00 as well as deprotecting the disk. When finished, using a sector editor, an additional 7 edits must be made on T$01, S$09 to disable the nibble count. See the following table:

Trk	Sct	Byte(s)	From	To	
$00	$02	$53	$96	$D5	
		58	D3	AA	DOS marker bytes
		5D	E5	AD	
		9E	96	DE	(edited by Super IOB)
		A3	FF	AA	
		E7	96	D5	
		F0	C5	C9	
		F1	31	AA	
		FC	E5	AD	
$00	$03	$35	$96	DE	
		3F	FF	AA	
		54	C5	C9	
		55	31	AA	
		5F	96	AA	
		6A	F2	96	
		90	C5	C9	
		91	31	DE	
		9B	B2	AA	
$00	$06	$7A	D3	D5	
		7F	96	AA	
		84	F2	96	
		AE	D3	DE	
		B3	B2	AA	
$00	$04	$29	$D5	$96	Translate Tables
		48	AA	D3	$BA29-$BA68 (Write)
		AA	1F	AA	$BA96-$BAFF (Read)
		D5	00	D5	(edited by Super IOB)
$01	$09	$73	$4C	$EA	Abort nibble count
		74	00	EA	
		75	C6	EA	(Sector edit these)
		81	D0	EA	
		82	0C	EA	
		89	D0	EA	
		8A	04	EA	

Controller

```
1000 REM MING CONTROLLER
1010 TK = 0:LT = 1:ST = 15:LS = 15:CD
     = WR:FAST = 1
1020 GOSUB 490: GOSUB 610
1025 RESTORE :T1 = TK:TK = PEEK (TRK):
     GOSUB 310:TK = T1
1030 GOSUB 490: GOSUB 610: IF PEEK
     (TRK) = LT THEN 1050
1040 TK = PEEK (TRK):ST = PEEK (SCT):
     GOTO 1020
1050 TK = 1:LT = 35:ST = 15:LS = 15:CD
     = WR:FAST = 1
1060 POKE 47426,24: POKE 47406,0: POKE
     47411,106
1070 RESTORE : GOSUB 190: GOSUB 210:
     GOSUB 170: POKE 47786,31: POKE
     47829,0
1080 GOSUB 490: GOSUB 610
1090 GOSUB 230: POKE 47786,170: POKE
     47829,213
1100 GOSUB 490: GOSUB 610: IF PEEK
     (TRK) = LT THEN 1120
1110 TK = PEEK (TRK):ST = PEEK (SCT):
     GOTO 1070
1120 HOME : PRINT "COPY~DONE" : END
5000 DATA 211,150,242
5010 DATA 150,211,229
5020 DATA 211,178,150,255
5030 DATA 27~CHANGES
5040 DATA 0,2,83,213,0,2,88,170
5050 DATA 0,2,93,173,0,2,158,222
5060 DATA 0,2,163,170,0,2,231,213
5070 DATA 0,2,240,201,0,2,241,170
5080 DATA 0,2,252,173,0,3,53,222
5090 DATA 0,3,63,170,0,3,84,201
5100 DATA 0,3,85,213,0,3,95,170
5110 DATA 0,3,106,150,0,3,144,201
5120 DATA 0,3,145,222,0,3,155,170
5130 DATA 0,6,122,213,0,6,127,170
5140 DATA 0,6,132,150,0,6,174,222
5150 DATA 0,6,179,170,0,4,41,150
5160 DATA 0,4,72,211,0,4,170,170
5170 DATA 0,4,213,213
```

Checksums

1000-$356B	1100-$E077	5080-$BDD9
1010-$EA41	1110-$32A4	5090-$6798
1020-$3164	1120-$1B51	5100-$9906
1025-$A976	5000-$FDC4	5110-$BADD
1030-$E423	5010-$381C	5120-$DEA8
1040-$4642	5020-$01D4	5130-$D843
1050-$E17B	5030-$2FA3	5140-$65AE
1060-$2267	5040-$E27D	5150-$883B
1070-$2D08	5050-$AAF5	5160-$C467
1080-$F980	5060-$0066	5170-$B4BC
1090-$1102	5070-$7BC4	

Softkey for...

Jigsaw
MicroLab

Jigsaw has a similar protection to that of Ming's Challenge. It has protected tracks $T01-T$09 with the same changed prologues and epilogues. However, besides T$00, T$0A-T$17 are normal and copyable with any sector copier. T$18-T$22 are unformatted. The translate table is likewise altered for the protected tracks, but there seems to be no nibble count to worry about. MicroLab has not bothered to compare nibbles with the value at $31 in this disk, so a controller can be easily written, and all sector edits can be made by Super IOB. Here are the softkey steps:

1. Install the controller listed, CON.JIGSAW, into Super IOB and run the resulting program. Answer "Y" when asked whether to format the disk.

2. The following sector edits are performed by Super IOB:

Trk	Sct	Byte(s)	From	To	
$00	$02	$E7	$96	$D5	
		F1	D3	AA	DOS marker bytes
		FC	E5	AD	
$00	$03	$35	$96	$DE	
		3F	FF	AA	
		55	D3	D5	
		5F	96	AA	
		6A	F2	96	
		91	D3	DE	
		9B	B2	AA	
$00	$04	$29	$D5	$96	
		48	AA	D3	Translate Tables
		AA	1F	AA	
		D5	00	D5	

Controller

```
1000 REM JIGSAW CONTROLLER
1010 TK = 0:LT = 1:ST = 15:LS = 15:CD
    = WR:FAST = 1
1012 POKE 775,96: ONERR GOTO 550
1015 GOTO 1020
1018 TK = 10:LT = 24:ST = 15:LS =
    15:CD = WR:FAST = 1
1020 GOSUB 490: GOSUB 610
1025 IF LT = 1 THEN RESTORE :T1 =
    TK:TK = PEEK (TRK): GOSUB 310:TK =
    T1
1030 GOSUB 490: GOSUB 610: IF ( PEEK
    (TRK) = LT) AND (LT = 1) THEN 1018
1035 IF PEEK (TRK) = LT THEN 1050
1040 TK = PEEK (TRK):ST = PEEK (SCT):
    GOTO 1020
1050 TK = 1:LT = 10:ST = 15:LS = 15:CD
    = WR:FAST = 1
1060 POKE 47426,24: POKE 47406,0: POKE
    47411,106
1070 RESTORE : GOSUB 190: GOSUB 210:
    GOSUB 170: POKE 47786,31: POKE
    47829,0
1080 GOSUB 490: GOSUB 610
1090 GOSUB 230: POKE 47786,170: POKE
    47829,213
1100 GOSUB 490: GOSUB 610: IF PEEK
    (TRK) = LT THEN 1120
1110 TK = PEEK (TRK):ST = PEEK (SCT):
    GOTO 1070
1120 HOME : PRINT "COPY~DONE" : END
5000 DATA 211,150,242
5010 DATA 150,211,229
5020 DATA 211,178,150,255
5030 DATA 14~CHANGES
5040 DATA 0,2,231,213,0,2,241,170
5050 DATA 0,2,252,173,0,3,53,222
5060 DATA 0,3,63,170,0,3,85,213
5070 DATA 0,3,95,170,0,3,106,150
5080 DATA 0,3,145,222,0,3,155,170
5090 DATA 0,4,41,150,0,4,72,211
5100 DATA 0,4,170,170,0,4,213,213
```

Checksums

1000-$356B	1050-$083E	5020-$963D
1010-$EA41	1060-$6704	5030-$0882
1012-$3D28	1070-$D811	5040-$A4E9
1015-$E82D	1080-$8CE3	5050-$1CD7
1018-$6D0A	1090-$58F5	5060-$60FE
1020-$DE79	1100-$F384	5070-$9DFD
1025-$847B1	1110-$A54D	5080-$7E0D
1030-$BBCC	1120-$0296	5090-$CA91
1035-$0D66	5000-$6A2D	5100-$DED1
1040-$0FB5	5010-$7D7F	

Jim S. Hart _____ NC

Softkey for...

Math Blaster Mystery

Davidson

Requirements:
4 blank disk sides
Whole disk copy program that will ignore read errors
ProDOS 8 and BASIC.SYSTEM
Optional - for putting all four sides on one 3.5" disk
Disk searcher/block editor
ProDOS utility program

Math Blaster Mystery is a very good educational program. Students are taught the thought processes to go through in solving math problems. Once a student becomes well versed in such processes, the intermediate problem solving steps can be passed over, and the student can concentrate on answering the questions. All in all, this 4 disk sided program is a quality product. Of course, it's copy protected. The copy protection involves an 'unformatted track', as well as nibble counting. Cookbook instructions are at the end of the article for the impatient.

The Protection

Attempting to fast copy the four sides using the Locksmith fast copy program reveals that all sides are normal except for side one. Track $22 of side one is 'unformatted'. Hmmm. It's very unusual for a non-copy protected disk to have an unformatted track, so I was lead to believe the disk was copy protected. I tried to boot up the copy of the first disk (the side with the unformatted track). The ProDOS copyright message came up and then a second later I was staring at the ProDOS quit code. The ProDOS quit code is a section of code that resides within ProDOS to allow a standard exit from all ProDOS programs. The one normally built into ProDOS is very unwieldy, hence the proliferation of program selectors like Bird's Better Bye or Squirt. These two programs replace the quit code and are mini program selectors. I have Bird's Better Bye installed in all of my ProDOS' - it's that good. I

recommend buying a Beagle Bros product just so you can get it. Anyway, I was staring at the original ProDOS quit code. A program does not normally do this unless maybe a copy protection scheme was not satisfied? Well, that turns out to be the reason. When a ProDOS disk is booted, the first two blocks on the disk get loaded in (boot blocks). After they are loaded and control passes to them, they immediately look for a file named PRODOS in the root directory. PRODOS is a system file (type SYS), and it's the ProDOS operating system. Once PRODOS has been loaded and control passes to it, it does some initialization, prints the ProDOS copyright message, and then scans the root directory for a type SYS file whose file name ends with '.SYSTEM' (i.e. BASIC.SYSTEM, UTIL.SYSTEM, etc). If it does not find one, an error message is displayed. If it does find at least one, ProDOS will then load and execute the first file in the root directory with the '.SYSTEM' extension. For example, suppose you have a disk with these files on it:

PRODOS	32	SYS
DUMMY.SYSTEM	2	BAS
COMPUTIST	99	SYS
BASIC.SYSTEM	20	SYS
UTIL.SYSTEM	56	SYS

When booted, PRODOS gets loaded first. ProDOS then runs BASIC.SYSTEM because it is the FIRST SYS file that ends with '.SYSTEM'. Looking at a catalog of the first side of Math Blaster Mystery, there is only one other SYS file in the root directory besides PRODOS: PS.SYSTEM. Guess which file gets run first after PRODOS? Remember when I booted the copy earlier and it dumped me into the quit code right after the ProDOS copyright message? That means the first '.SYSTEM' file had to have done it, which means the protection code is in the first '.SYSTEM' file. This is where I booted a ProDOS disk with BASIC.SYSTEM on it and got into BASIC. I inserted side one of the copy and set the prefix to it:

PREFIX /MBM

Since the PS.SYSTEM file had the protection code in it, I loaded the file to look it over:

BLOAD PS.SYSTEM, A$2000, TSYS

SYS files ALWAYS load in at $2000. Guess where I started looking? Here is a bit of the code there:

2000L

```
2000: 20 D2 25  JSR $25D2
2003: 4C 74 08  JMP $0874
2006: 8E 04 C0  STX $C004
2009: 8E 02 C0  STX $C002
```

A subroutine is executed at $25D2 (Jump to SubRoutine at) and then control passes to $0874 via a JuMP statement. Since SYS files start loading at $2000, I found it odd that a jump is made to $0874. The subroutine at $25D2 must have a memory move in it somewhere, and it must move memory to around the $0874 area. Let's take a look at the subroutine at $25D2:

25D2L

```
25D2: A2 00     LDX #$00
25D4: BD 03 20  LDA $2003
25D7: 9D 01 08  STA $0801
25DA: BD 03 21  LDA $2103
25DD: 9D 01 09  STA $0901
25E0: BD 03 22  LDA $2203
25E3: 9D 01 0A  STA $0A01
25E6: BD 03 23  LDA $2303
25E9: 9D 01 0B  STA $0B01
25EC: BD 03 24  LDA $2403
25EF: 9D 01 0C  STA $0C01
25F2: BD 03 25  LDA $2503
25F5: E8        INX
25F6: D0 DC     BNE $25D4
25F8: 60        RTS
```

Well, we've found the memory move. Since it is a stand alone subroutine, let's execute it so the code that ends up at $874 will be moved to $874:

25D2G

The code has been moved, so let's take a look to see what is at $874:

874L

```
874: 20 C8 0A  JSR $0AC8
877: F0 D9     BEQ $0852
879: AD 98 BF  LDA $BF98
```

The first thing to notice is that a subroutine call to $0AC8 is executed. Right after that, the zero flag is checked. If there was a zero result in the subroutine, the BEQ (Branch on EQual result) will be taken and control will transfer to $0852. A look at the code at $0852 reveals:

0852L

```
0852: 8E 04 C0  STX $C004
0855: 8E 02 C0  STX $C002
0858: 20 00 BF  JSR $BF00
085B: 65
085C: 5E 08
```

Ah ha! This is where reference material like Beneath Apple ProDOS and back issues of Computist come in handy. The JSR $BF00 is a ProDOS machine language interface (low level operating system) call. The first byte after it is the type of call (65 = QUIT CODE!) and the next two bytes are the location of the call parameters.

What's important is that we have just found the quick quit code call I was experiencing on the copied disk. We don't want the quit code to be called (failed protection check = quit code call). Where was the quit code called from? Well, remember the code that started at $874? Notice the BEQ $0852? We DO NOT WANT this branch to be taken. Taking that branch means that the protection code at $0AC8 failed. Now, how do we circumvent it? There are a number of ways. We could taken out the call to the protection (JSR $0AC8), follow the protection code and alter it so it always reports a successful check, or simply get rid of the branching command (BEQ $0852). I decided to take the simple way out and NOP the branch. I changed the code at $0874 to this:

877:EA EA (was F0 D9)

874L

```
0874: 20 0A C8  JSR $0AC8
0877: EA        NOP (was F0)
0878: EA        NOP (was D9)
```

NOPs in machine language have the same function as REMs in BASIC: do nothing. I made this change and saved the file back to the copied disk:

BSAVE PS.SYSTEM,A$2000,TSYS

Booting the first side now results in no quit code and a functioning program. The protection check has been taken care of.

Putting it all on one 3.5" disk

One final thing bothered me. The 4 floppy disks are in ProDOS format, so why couldn't they all be put on one 3.5" disk? Trying to do so will initially give you nothing but frustration. There is a free upgrade offer included with the package, but why wait? If we can deprotect it, we can fix it to run on a 3.5" disk. The problem that the program uses absolute prefixes instead of relative ones. For example, it expects to find a disk named '/MBM2' when it needs the files from side two. Now, if we could change it so that instead of looking for a DISK named '/MBM2', it would look in a SUBDIRECTORY named AMBM2, our problems would be solved. Why use 'AMBM2' in place of '/MBM2'? AMBM2 has as many characters as /MBM2 and to convert /MBM2 to AMBM2 only requires that you change the '/' to a 'A'. Boot up your disk searcher/block editor (I used Copy II Plus) and read block $7 of the copied first side of disk one. Now search for the HEX sequence:

4D 42 4D 32

Those are the hex values for the characters 'MBM2'. When you find them, you will also see the other disk's names and subdirectory names, for example:

/MBM2 / . /MBM3 / . /MBM4 .
/MBM2/DATA. /MBM4 /DATA.

or something very similar. These are the pathnames the program uses to load its files. Here is where we make our changes. For example, if we change '/MBM2/' to 'AMBM2/', then the program will look in a subdirectory 'AMBM2/' instead of a disk '/MBM2/' for its files. The steps to put all of the floppies onto one 3.5" disk are in the cookbook section. This procedure can be applied to many other ProDOS based programs that have their pathnames hard coded into them. One other program this works for is the ProDOS version of Sensible Speller. Enjoy!

Part 1 - The softkey

1. Fast copy all four original disk sides onto the four blank sides. Ignore errors on disk #1, side #1.

2. Boot up ProDOS and get into BASIC.

3. Insert disk #1, side #1 into a disk drive and type:

PREFIX /MBM

4. Load up the main system file:

BLOAD PS.SYSTEM, A$2000, TSYS

5. NOP out the branch:

2079:EA EA *was F0 D9*

6. Save the deprotected file back to disk:

BSAVE PS.SYSTEM, A$2000, TSYS

The deprotection is complete!

Part 2 - Putting it all on a 3.5" disk

1. Boot up your ProDOS utility program (I used Copy II Plus).

2. Format a 3.5" disk and name it /X.

3. Copy all of the files from disk #1, side #1 onto /X.

4. Make four subdirectories on /X and name them

AMBM2
AMBM3
AMBM4

A catalog of the disk will look like this:

```
/X————AMBM2
|
|————AMBM3
|
|————AMBM4
```

5. Copy the files from disk #1, side #2 to /X/ AMBM2/.

6. Copy the files from disk #2, side #1 to /X/ AMBM3/.

7. Copy the files from disk #2, side #2 to /X/ AMBM4/.

8. All of the files from the first side should be in the main directory, and all of the files from the other sides should be in their own subdirectories now. Boot your disk searcher/block editor.

9. Read in block $7.

10. Search the disk for the HEX sequence 4D 42 4D 32. When you find it, you should see the names of the other disks in the ASCII section of your block editor. Make the following changes:

CHANGE	TO
2F 4D 42 4D 32	41 4D 42 4D 32
2F 4D 42 4D 33	41 4D 42 4D 33
2F 4D 42 4D 34	41 4D 42 4D 34

You are changing the '/' in front of the disk names into the letter 'A'. Don't forget to change ALL occurrences of the above bytes in the block. There will be two occurrences of the first and third sequence of bytes.

11. Write the block back to disk.

Your Math Blaster Mystery program is now all on one 3.5" disk! To speed things up even more on a GS, boot the disk after you've installed Diversi-Cache. It speeds disk accessing by 300% or more.

Softkey for...

California Games GS

Epyx

Requirements:
A blank formatted 3.5" disk
3.5" disk copy program
Block editor/disk searcher

I tried out Dan Halfwit's California Games GS softkey in Computist #64 and it didn't work all the way for me. I checked the block he spoke of ($FC) and compared his 'TO' bytes to what was on my disk. It seems some of the code moved around a little, but the protection still is there.

1. Copy the original disk onto the blank formatted 3.5" disk.

2. Using your block editor, make the following changes to the copied disk:

Blk	Byte(s)	From	To
$FC	$66	22	AF
	$77-78	90 03	80 03
	$88	22	AF
	$9C	90 03	80 03
	$B9	90 1D	80 18

3. Write the block back to disk and you're done!

The Protection Scheme

Below is a partial listing of the protection scheme. I've sent it in so readers can have something to look at and compare to when they go about looking for a GS protection scheme.

[Block $FC]

```
115A: A2 20     LDX #20    ;track $20
115C: A0 01     LDY #01    ;side 1
115E: 5A        PHY
115F: DA        PHX
1160: F4 00 00  PEA 0000
1163: F4 AA 52  PEA 52AA
1166: 22 93 53 00 JSL 005393 ;call nibble count routine
116A: 8E 9E 52  STX 529E
116D: 8C 9F 52  STY 529F
1170: A8        TAY
1171: 68        PLA
1172: 68        PLA
1173: 68        PLA
1174: 68        PLA
1175: 68        PLA
1176: 68        PLA
1177: 90 03     BCC 117C   ;good nibble count if
                            taken
1179: 4C 93 52  JMP 5293   ;crash
117C: A2 21     LDX #21    ;track $21
117E: A0 01     LDY #01    ;side 1
1180: 5A        PHY
1181: DA        PHX
1182: F4 00 00  PEA 0000
1185: F4 AA 52  PEA 52AA
1188: 22 93 53 00 JSL 005393 ;call nibble count routine
118C: 8D D0 52  STA 52D0
118F: 8E A4 52  STX 52A4
1192: 8C A5 52  STY 52A5
1195: A8        TAY
1196: 68        PLA
1197: 68        PLA
1198: 68        PLA
1199: 68        PLA
119A: 68        PLA
119B: 68        PLA
119C: 90 03     BCC 11A1   ;good nibble count if
                            taken
119E: 4C 93 52  JMP 5293   ;crash
11A1: C2 30     REP #30
11A3: 22 76 53 00 JSL 005376
11A7: 28        PLP
11A8: FB        XCE
11A9: 28        PLP
11AA: 08        PHP
11AB: 18        CLC
11AC: FB        XCE
11AD: 08        PHP
```

```
11AE: C2 30      REP #30
11B0: A0 FF FF    LDY #FFFF
11B3: AD 9E 52    LDA 529E
11B6: CD A0 52    CMP 52A0
11B9: 90 1D       BCC 11D8    ;bad
11BB: AD 9E 52    LDA 529E
11BE: CD A2 52    CMP 52A2
11C1: B0 15       BCS 11D8    ;bad
11C3: AD A4 52    LDA 52A4
11C6: CD A6 52    CMP 52A6
11C9: 90 0D       BCC 11D8    ;bad
11CB: AD A4 52    LDA 52A4
11CE: CD A8 52    CMP 52A8
11D1: B0 05       BCS 11D8    ;bad
11D3: 28          PLP
11D4: FB          XCE
11D5: 28          PLP
11D6: 18          CLC
11D7: 60          RTS
11D8: C2 30       REP #30     ;come here if count was
                                 bad
```

The changes I made were:

original			from:to	
1166: 22 93 53 00	JSL 005393		22:AF	LDA
1177: 90 03	BCC 117C		90:80	BRA
1188: 22 93 53 00	JSL 005393		22:AF	LDA
119C: 90 03	BCC 11A1		90 80	BRA
11B9: 90 1D	BCC 11D8		90:80	BRA

The changes at 1166 and 1188 disabled the calls to the nibble counts. The changes at 1177 and 119C jumped over a crash routine call. Finally, the change at 11B9 jumps over a whole bunch of checks and just goes to where the code would go if the nibble count had been successful.

A Collection of short softkeys

Softkey for...

Antonyms & Sentence Completion (Best sentences)
Queue Intellectual Software
Analogies - Advanced I & Advanced II
Hartley

Requirements:
1 blank disk
Super IOB v1.5
New Swap controller
RWTS WORM from Computist #61

1. Boot normal DOS 3.3 and load up the RWTS WORM:

BLOAD RWTS WORM,A$9500

2. Insert original disk into slot 6 drive 1 and execute the WORM:

CALL 38144

3. After a few seconds the RWTS will be in memory at $1900. Take out the original disk, insert your Super IOB disk, and save the RWTS:

BSAVE RWTS.rwts name, A$1900, L$800

4. Install the New Swap controller into Super IOB and then make the following change:

```
1015 TK = 3
10010 PRINT CHR$(4) "BLOAD RWTS.rwts
   name, A$1900"
```

5. RUN the controller and when it is done, your disk will be deprotected.

Softkey for...

Reading Professor
Executive Systems, Inc.

Requirements:
3 blank disks
COPYA
Sector editor

1. Boot DOS 3.3, disable error checking, and run COPYA:

POKE 47426,24
RUN COPYA

2. Copy all three disks onto the blanks.

3. On the Reading Professor disk, make the following sector edits so that the disk can read itself:

Trk	Sct	Byte(s)	From		To	
$00	$03	36-37	D0 0A		18 EA	
		92-93	D0 0A		18 EA	

4. Write the sector back to disk and you are done!

The edits (and the POKE 47426,24) were necessary because some of the sectors on track $03 are different from standard DOS 3.3.

Softkey for...

Word Puzzles for Creative Teaching
EBSCO Curriculum Materials

Requirements:
Blank disk
Super IOB
New Swap controller
RWTS WORM from Computist #61
Fast DOS 3.3 (optional)

On the original disk's sleeve, there is a label

I thought you might be interested in reading: "This disk is copy protected. Attempts to copy will cause data to be altered, and will void replacement guarantee"

Well, here is a better replacement 'guarantee': Computist! Follow the steps below to guarantee you won't have to call EBSCO for a replacement disk:

1. Boot up DOS 3.3 (a fast DOS is preferable) and initialize the blank disk:

INIT HELLO

2. Load up the RWTS WORM:

BLOAD RWTS WORM, A$9500

3. Insert the original disk and activate the WORM:

CALL 38144

4. Insert your Super IOB disk and save the RWTS:

BSAVE RWTS.EBSCO, A$1900, L$800

5. Install the New Swap controller into Super IOB and make the following changes:

```
1015 TK = 3
10010 PRINT CHR$(4) "BLOAD RWTS.EBSCO,
   A$1900"
```

6. RUN the controller to copy the original onto the newly formatted disk.

You're done! Now, you don't have to live with EBSCO's 'guarantee' any more.

Softkey for...

Microcomputer Study Guide - Fundamental Accounting Principles, Chapters 1-14 & 15-28
Richard D. Irwin, Inc.

Requirements:
4 blank disks (or 2 disks that are double sided)
Whole disk copier
Disk searcher/sector editor

The protection on these four sides is the exact same as on KC's Deals On Wheels (softkey in Computist #46). Refer to that article for an explanation.

1. Use your whole disk copy program to copy all four sides onto the blank four sides.

2. Boot up your disk searcher/sector editor and search for 2C E9 C0, replace with 18 90 5A. Make the changes to each of the four sides. There are two such edits to be made on each disk side, and they are usually within a track of each other. Mine were only $0F and $10, but yours may be different, so scan for them.

3. Write the changed sectors back to disk and you're done!

Softkey for...

Multiscribe v3.0
Claris Corp.

1. Copy the original disk with any whole disk copier.

2. Scan the copied disk for A2 00 B5 00 9D 00 B A CA D0 F8 and change it to A9 00 18 60 9D 00 BA CA D0 F8.

3. Write the change back to disk and you're done.

Softkey for...

Starglider
Rainbird/Firebird Software

Requirements:
A blank 5.25" disk
Whole disk copy program such as COPYA
Sector editor/disk searcher
A fast DOS (optional)

Removing the manual check

In Computist #64, Edward Croft Jr. had a softkey for Starglider. I thought back because I had cracked this program a long time ago and had (gasp) forgotten to send the crack in to Computist! Well, my way is different from Ed's in the sense that it removes the manual check so that it doesn't even ask you any more.

1. Copy the original Starglider disk onto the blank disk using your whole disk copy program. To speed things up, copy a fast DOS onto the first three tracks.

2. Get out your sector editor and make the following edit to your copied Starglider disk:

Trk	Sct	Byte(s)	From		To	
$0F	$0F	A0-A2	20 00 60		EA EA EA	

3. Write the sector back to disk. You're done!

The Protection Scheme

Below is a listing of the code around the call to the password protection in case yours is in a different spot. I found the code by following the files in the order that they were executed.

Track $0F, sector $0F

```
008C: 20 28 42    JSR 4228
008F: C9 01       CMP #01
0091: D0 03       BNE 0096
0093: 20 E6 41    JSR 41E6
```

```
0096: 18          CLC
0097: 69 01       ADC #01
0099: C9 04       CMP #04
009B: D0 0F       BNE 00AC
009D: 8D 8B 41    STA 418B
00A0: 20 00 60    JSR 6000    password check
00A3: 20 D9 41    JSR 41D9
00A6: 20 E6 41    JSR 41E6
00A9: AD 8B 41    LDA 418B
00AC: C9 07       CMP #07
00AE: 90 DC       BCC 008C
00B0: 60          RTS
```

Softkey for...

Children's Writing and Publishing Center
The Learning Company

Requirements:
Original disks
A blank disk for each side
COPYA from Apple DOS 3.3 System Master disk

1. Boot up DOS 3.3.

POKE 47426,24
RUN COPYA

2. Copy all disk sides.

3. On the copy of side #1, load the file with the protection scheme:

BLOAD RDOS.AUX
7764:4C F7 77 was A5 DA F0, skip protection
BSAVE RDOS.AUX,A$

You're done.

To Charles Haight: I agree with you on the tabloid format of Computist. It would be fine to receive Computist in that form. The only complaint I have with it is that the front and back covers are exposed. Adding a blank cover page to the front and back wouldn't add any more to the cost and it would protect the magazine better. OPEN-Apple, a monthly magazine from TechAlliance, puts a cover page on both the front and back of their magazine and it really is a nice touch. As for the costs of typesetting equipment, let me pose a possible solution. This is a computer magazine geared primarily towards Apple computers, correct? Why not produce the magazine using a Macintosh? Let's see....$2300 a month for 6 months comes out to $13800. That's more than enough to buy a Macintosh system complete with Laserwriter printer. You could completely produce and print the master copy of the magazine "in house". When it comes time to make the copies for the subscribers, the magazine would already be in photo-ready format. You wouldn't need to lease the typesetting equipment anymore which would save money.

Another potential money making venture would be a creation of a book I term "THE COMPUTIST LOG". What this book would consist of are all of the non-deprotection related articles and programs that have been in Computist in past issues. You could even have more than one book. For example, in the Computist Log you could have the articles on modifying the printer drivers in Print Shop & Newsroom, map editors/printers for various games such as mine for Ultima IV, game playing tips, etc. The Log could be divided up into chapters with different chapters each having having a different focus, like the above mentioned articles and programs. There could be a chapter (a large one) devoted to nothing else but game tips and hints. Another chapter would contain APTs. Another would contain programs. I think this would be a sought after book once the general computer public were made aware of it. I know I'd buy one of the books to have all of the info in one easy to use reference.

To Gary Verbuch: There definitely are LISP compilers for the Apple // series of computers. Two such compilers are App-L-ISP ($50) from Special Software Systems and KeyLISP ($149) from XPrime Corporation. The addresses for the companies are:

Special Software Systems
1655 Campbell Ave.
Thousand Oaks, CA 91360
(805) 494-0040

Xprime Corporation
10835 Santa Monica Blvd., Ste. 204A
Los Angeles, CA 90025
(213) 470-4663

I suggest you join TechAlliance (formerly Apple Programmers and Developers Association). If you are a member, you can get a free guide to languages and tools available for the Apple // and Macintosh computers. There is a lot more out there, compiler wise, for the Apple // series than I suspected.

To Rich Linville: There are several ways to put a DOS 3.3 program from a 5.25" disk onto a 3.5" disk. One choice is to format the 3.5" disk under ProDOS and then copy PRODOS and BASIC.SYSTEM to the disk. Now use your ProDOS disk utility program (I use Copy II Plus) to copy all of the files from the 5.25" disk to the 3.5" disk. This will work quite well if the copied files are all Applesoft. Another choice is to use a modified DOS 3.3. UnoDOS, AMDOS, and UniDOS 3.3 are all DOS 3.3s that have been modified so that they will run on a 3.5" disk. Once

---

you have formatted the 3.5" disk with one of these DOSs, use the supplied copy program to copy the files from the 5.25" disk to the 3.5" disk. There are some programs, however, that are made to run on 5.25" disk and no other. Usually these are programs that directly access the slot 6 ROM to improve disk access time.

Various Thoughts: GBBS would be, in my opinion, the best BBS software to get for the Computist BBS. There are thousands of BBSs running off of it and there is tremendous support for it.

I just bought a little graphics package called Spectragraphix 256. It kind of resembles Iconix in what it does. It allows you to use Applesoft BASIC to draw onto the super hi-res screen, animate shapes, and put text anywhere on the super hi-res screen. The program itself is really good, but the documentation leaves a lot to be desired. A revised manual with lots more commented examples would make this a first rate package.

TML Systems will be releasing TML Pascal II around the first of July 89. That is when Apple will be releasing the GS System Disk v5.0. One intriguing feature of the new language system is the inclusion of a Resource Editor. No longer will GS programmers have to deal with cryptic menu strings and the like. The resource editor will allow the programmer to design on screen what the menus, windows, and general operating environment will look like by a simple point-and-click method: WYSIWYG programming! The upgrade for registered TML Pascal and TML BASIC owners is $49.95, and the new Source Code Library II is available for $29.95. GS programmers should look into this new package.

Softkey for...

Monsters and Make Believe
Pelican Software

This softkey is the same as the softkey for PIRATES GS in Computist #61, page 8.

Putting a deprotected Ultima I on a single disk side

I recently purchased the re-released version of Ultima I. The original was written in BASIC and after playing it for a while you really noticed the lack of speed. The re-release, however, has been completely rewritten in assembly language. No more annoying waits; everything moves crisply and quickly. I deprotected the disk so I could play with a backup. Another benefit I wanted was to put the program on a 3.5" disk, since it was ProDOS based, and to run it after Diversi-Cache had been installed so it would load super quick. Copying the files over to the 3.5" disk was no problem. Running it was no problem. The only problem came when I wanted to save a game. It would not save to the same disk; it had to save its games onto another disk. Now that turned out to be really annoying. Why couldn't I save the games on the same disk that contained the program files? Not enough space? Wrong. The saved game file is only a block or two long; it could fit with no trouble on the 5.25" disk the original copy was on. It turned out that the problem was pathnames.

The program disk for Ultima I is named /U1 and the data disk is named /U1.PLAYER. Since the program and data disk names were not the same, I couldn't just use one disk to play the game unless I modified the code. Well, I modified the code! Looking through the program files, I came across the program and data disk names in ASCII format. The change I made was to change the data disk's pathname from /U1.PLAYER to /U1/PLAYER. I just changed the period between /U1 and PLAYER to a slash. The result of this is when you go to save a game now, the game is saved to the subdirectory PLAYER, which is located on the program disk. No more disk swapping and super fast loading on a GS with a 3.5" drive and Diversi-Cache. This is how gaming is supposed to go!

1. Deprotect your original Ultima I disk.

2. If you have a 3.5" drive, format a 3.5" disk and name it /A. Now copy all of the files from the deprotected copy to the 3.5" disk. After all of the files have been copied, rename the 3.5" disk to /U1.

3. Boot up ProDOS, get into BASIC, and create a subdirectory on the disk and call it PLAYER:

PREFIX /U1
CREATE PLAYER

4. Load up the file with the pathnames in it:

UNLOCK MI.U1
BLOAD MI.U1,A$7000

5. Go into the monitor and change the period in the data disk pathname to a slash:

CALL -151
7E3B:2F   *was 2E*

6. Save the modified file back to disk:

BSAVE MI.U1,A$7000
LOCK MI.U1

You're done! If the pathnames have been moved around in the file, scan the file for the byte sequence 55 31 2E 50 4C (hex values for U1.PL) and change the 2E to a 2F.

The Hardware Corner

Well here we are again, Hardware City. In part 1, I gave the combined schematic and an explanation of how it works. I also asked for comments on how well the explanation was written. Your response was underwhelming. We didn't even have enough interest to enable us to order the circuit boards. Only a few wrote and your comments were about evenly split between; the explanation was too indepth, the explanation was too shallow or who cares; print more softkeys. A throughly discouraging response and I hope it changes soon or I'm going to resign as your hardware guru and let someone else take over. I not only haven't made a goal but it seems that I'm not even on the playing field yet.

So here's part 2 for the handful of readers who are interested (thank goodness for you).

The Display Card (Part 2)

Here are the actual schematics, parts lists and board layouts for the Display card.

I've tried to give part numbers from the current Jameco catalog whenever possible. I've always bought my parts from Jameco because of their fast service, wide selection of parts and low prices. They do have a minimum order amount, however, so you might have to look elsewhere if you only need a few items.

Circuit card 1 plugs into a slot, while card 2 sits outside. A 14 pin DIP jumper connects the two cards. Figure 1 is the circuit board parts layout diagram for card 2. Figure 3 is it's schematic diagram. Figure 2 is the circuit board parts layout diagram for card 1. Figure 4 is it's schematic diagram.

This is a fairly simple project and none of the parts are critical. Feel free to substitute whenever necessary. If you have any problems, write to me in care of Computist. If anyone knows of a place where we can get the boards made at a reasonable cost please let me know.

Card 1 Parts List

Label	item	Jameco #
Cf	4.7 µf capacitor	R4.7/50
C1	.1 µf capacitor	DC.1
C2	.1 µf capacitor	DC.1
IC1	74LS521	74HCT688
IC2	74LS374	74HCT374
IC4	74LS02	74HCT02
IC5	74LS08	74HCT08
IC6	74LS74	74HCT74
IC7	555 timer	NE555V
IC8	555 timer	NE555V
R1	2.2 M 1/4 watt resistor	R2.2M
R2	220 K 1/4 watt resistor	R220K
R3	10 K 1/4 watt resistor	R10K
R4	10 K 1/4 watt resistor	R10K
R5	10 K 1/4 watt resistor	R10K
R6	10 K 1/4 watt resistor	R10K

Misc.

2 ea.	8 pin DIP socket	8LP
4 ea.	14 pin DIP socket	14LP
2 ea.	20 pin DIP socket	20LP
1 ea.	14 pin DIP to DIP jumper (24")	DJ16-2-16

Notes

P1 is a 14 pin DIP socket. The DIP jumper part number listed is for a DIP jumper with 16 pins, be careful that you connect it correctly to both boards. (Or you could find another source for a 14 pin DIP to DIP jumper.) Capacitor Cf is a filter capacitor that fits on the lower right corner of circuit board 1.

Card 2 Parts List

Label	item	Jameco #
C1	.01 µf capacitor	DC.01
C2	4.7 µf capacitor	R4.7/50
D1	1N914 small signal diode	1N4148
IC2	74LS138	74LS138
IC3	74LS138	74LS138
IC4	74LS138	74LS138
IC5	555 timer	NE555V
IC6	74LS21	74LS21
LA1	16 individual LEDs	XC57124R
LA2	5 individual LEDs	XC57124R
R1	1 K 1/4 watt Resistor	R1.0K
R2	500K Horiz mount trim Pot	3355-500K
R3	1/4 watt resistor	R470
R4	1/4 watt resistor	R470
SWT1	SPST switch (PC mount)	SW135
SWT2	push button switch	SW178
SWT3	SPDT switch (PC mount)	SW135
SWT4	SPST switch (PC mount)	SW135

Misc.

1 ea.	8 pin DIP socket	8LP
2 ea.	14 pin DIP socket	14LP
3 ea.	16 pin DIP socket	16LP

Notes

P1 is a 14 pin DIP socket. Use short lengths of wire to solder swt2 to the board, than use a dab of silicon seal to glue the switch to the board. The part numbers shown for the SPST switches are actually for DPDT switches. (They sold out of the SPST switches.) Clip the extra pins and they'll work just fine. If they're out of SW135 try SW127.
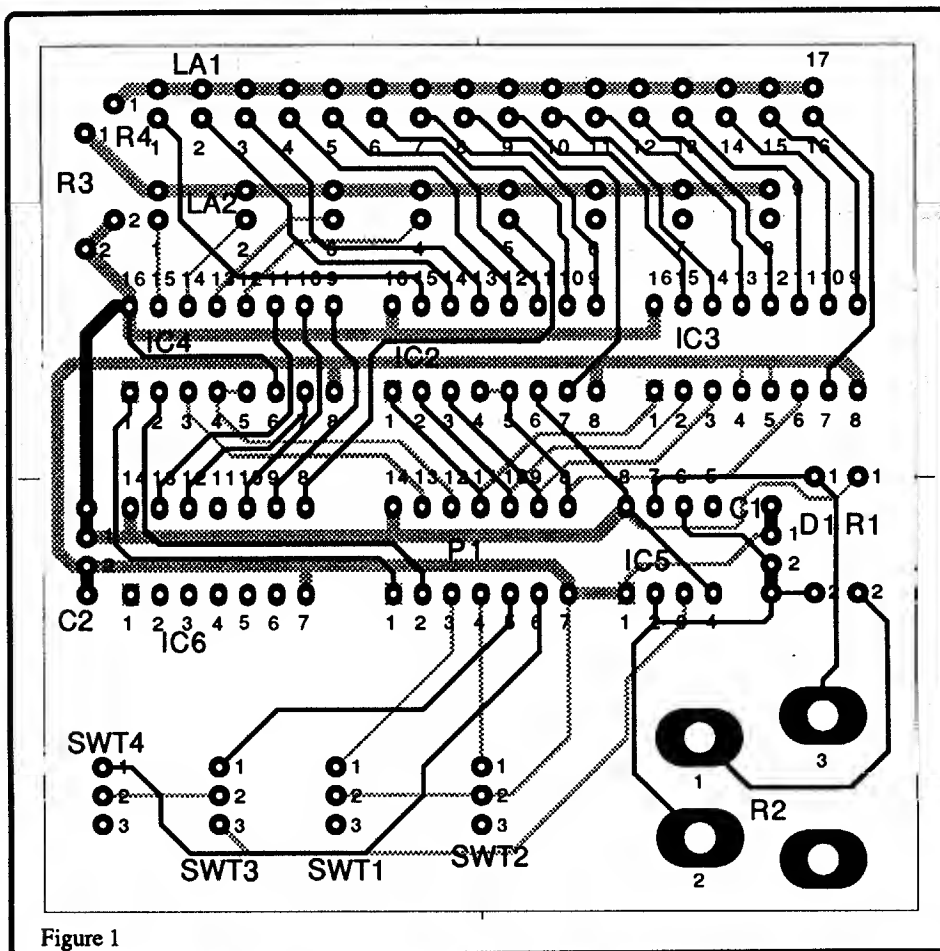
Figure 1
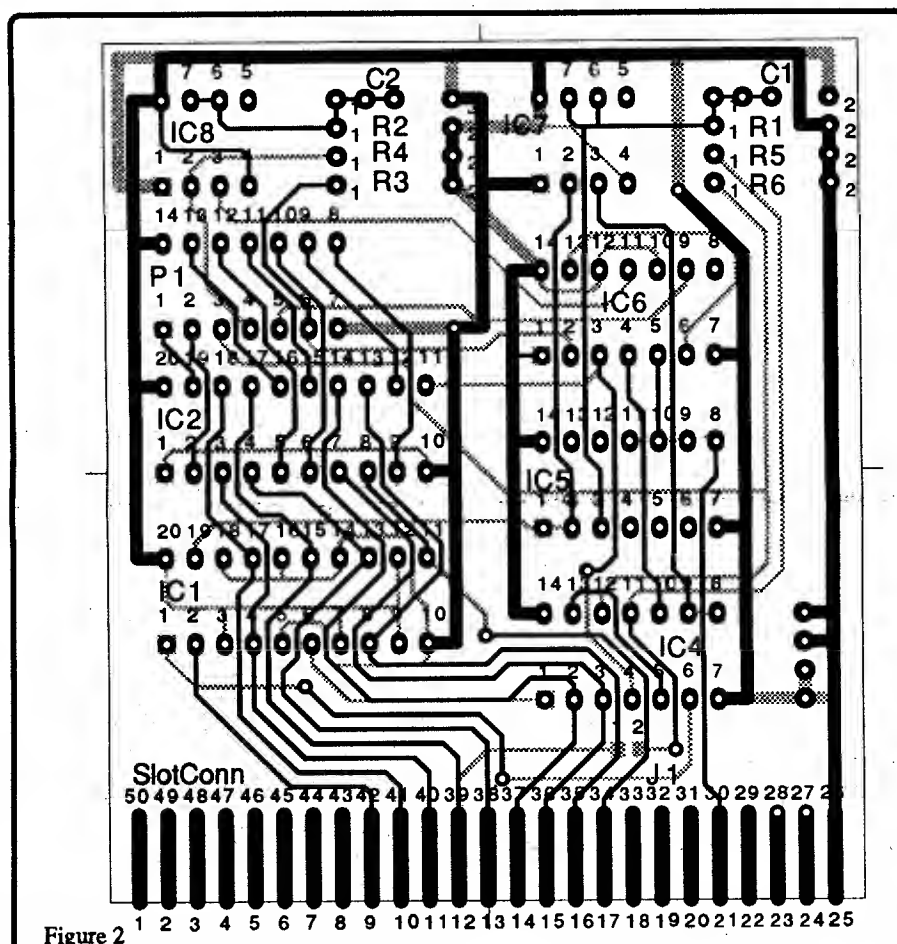
Figure 2
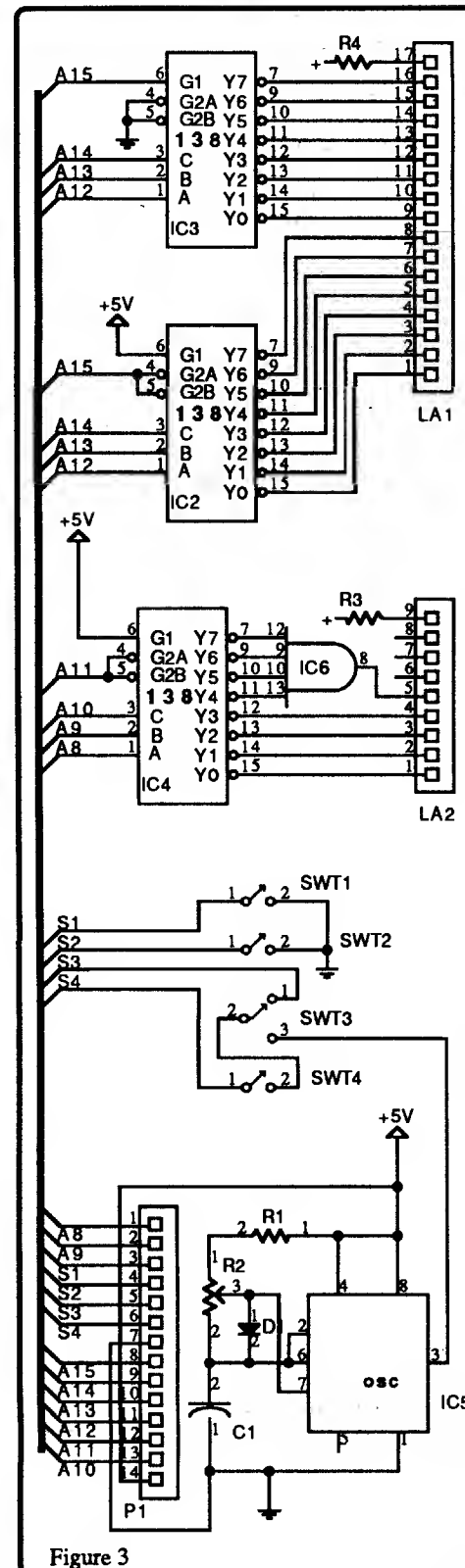
Figure 3

The circuit diagrams and board layouts were drawn using Douglas CAD. The layout files, parts lists and NET lists are available on 3.5" disk (Apple Mac format) for those of you who can make use of the information. Send $3 to cover the cost of the disk plus postage and handling.

I suppose I could even convert it to a standard text file as postscript code. I'll look into that a little more if it turns out that anyone is interested.

Figure 4

Marc Batchelor _____ FL
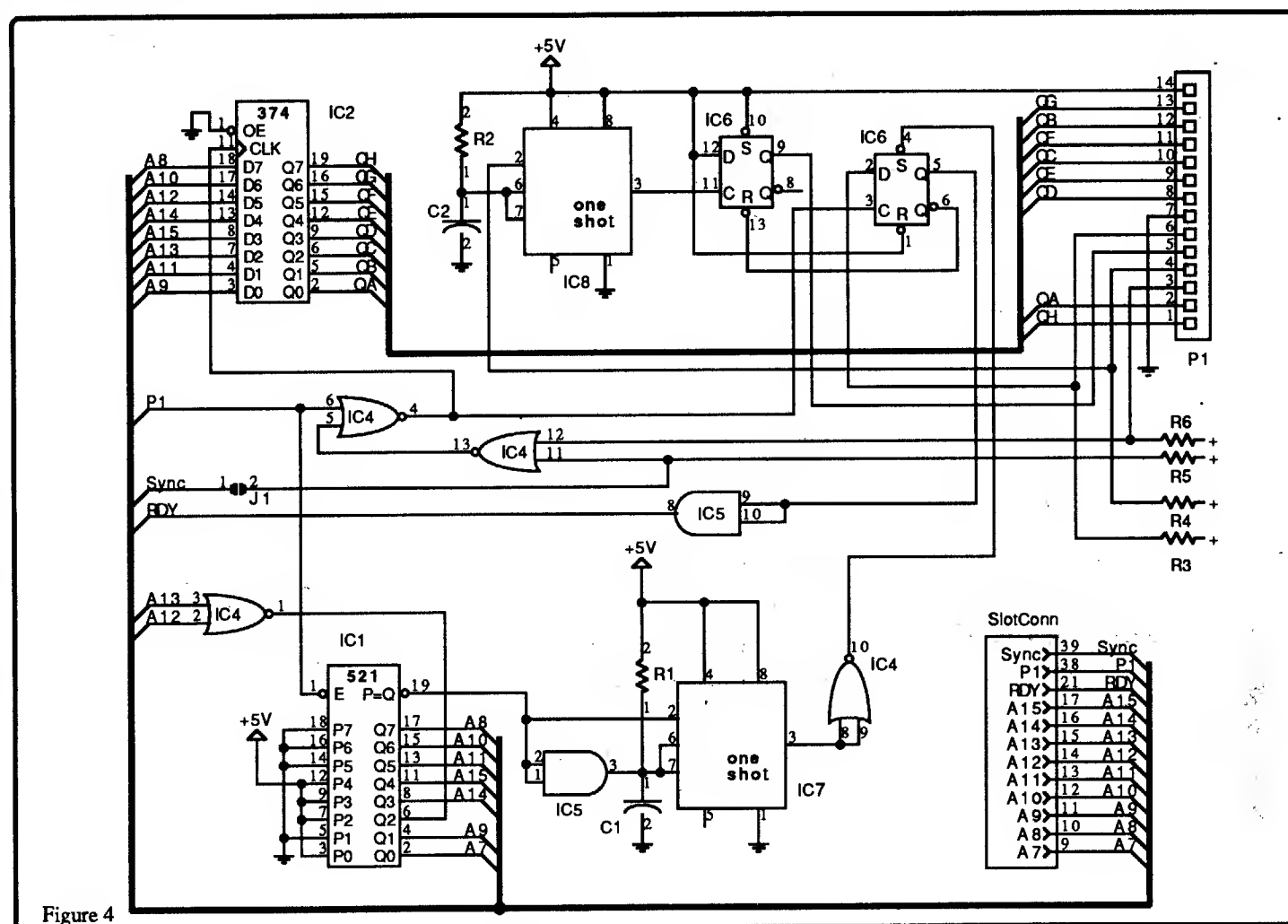
Softkey for...

Zany Golf

Electronic Arts

A friend recently asked me to come up with a crack to Zany Golf. Zany Golf (ZG) is protected with a code. That is, upon reaching the Hamburger Hole, you are faced with the following type of prompt:

ENTER CODE: Windmill 4 Iron Eagle:

You must look in the manual, locate this sequence and type the four character code associated with the above "clue". While this is effective in preventing casual "giving" of the program to friends, it does not do much in the way of protection since many career pirates will diligently type in code tables, code wheels, etc. and distribute the information on the disk as a textfile for printing. The code idea is also a hassle for registered owners and real users. However, I prefer this kind of "protection" to nibble checks and key disks. This is easily installable on a harddrive and is a joy to play.

The Protection:

The protection resides in the file ZANY.COM. If you DEBUG the program and search from 100 to FFFF for the "ENTER CODE", you will find that it spits out one address. Searching for references to the byte preceding this address (length byte) will turn up a reference in the $F500 page (for the benefit of true IBM'ers, '$' means HEX). I then traced the code all the way to it's conclusion where it either continued or performed a call to $1CA2, which sets the video mode and exits via INT 20. Looking at how the code branched, I found that $F754 was where the code branched too, on successful entry of the code. The test for this I found at $F736. The code looked like the following:

```
XXXX:F736 75 03      JNZ    F73B
XXXX:F738 E9 19 00   JMP    F754
XXXX:F73B            Crash Code
XXXX:F754            Start Hamburger Hole
```

To circumvent this check, I decided to overwrite some of the code in the $F500 page (before the ENTER CODE message was displayed).

S100 FFFF 75 03 E9 19 00 89 C2

Response S/B: XXXX:YYYY. Write down YYYY and use in next step.

UYYYY

Response S/B:

```
XXXX:YYYY 75 03      JNZ ????
XXXX:YYZZ E9 19 00   JMP AABB
```

Write down AABB to use later.

S100 FFFF B8 88 F7 A3

Response S/B: XXXX:CCCC. Write down CCCC and use in next step.

ACCCC

Response S/B: XXXX:CCCC

JMP AABB
NOP
press return here
W   _Writing XXXX Bytes._

You should be done. Now, the program will continue to load and execute the hamburger hole without asking you for the code. That's it.

Advanced Playing Technique for...

Zany Golf

Electronic Arts

For those interested in obtaining unlimited strokes, follow this procedure:

DEBUG ZANY.COM
S100 FFFF 8A 84 F4 59 3C 01 F5

```
XXXX:YYYY
UYYYY
MOV AL, [SI+ZZZZ]       Write down ZZZZ
CMP 01
CMC
AYYYY
XXXX:YYYY
MOV AL,05
MOV [SI+ZZZZ],AL   ZZZZ is obtained from above
RET
```

This provides unlimited strokes WITHOUT affecting total scoring. Therefore, if you do hit the ball 49 times on the energy hole, it shows up on your score card. Enjoy.

This information was sent to us on disk. It is from a BBS. The names were removed before we got them so we are unable to give the authors proper credit. If you recognize a procedure, please let us know. We would like to receive the info direct from you next time.
...RDEXed

Playing Tips for ...

Manhunter

?

video game	Find the shortest path; record t three dolls for Coney Island.
The sewer maze	The sewer maze is the same as the video game.
Seasons combinations	(4,1) (1,0,3,1) (2,6,4) (4,2,5).
Suspect names	Phillipe Cook, Harvey Osborne, Anna Osborne, Harry Jones.
MAD information	The names are: Harvey Osborne, Harry Jos, Phillipe Cook.
The church candles	Top Row: 1st candle; 2d Row: 3rd candle; 3rd Row: 4th candle.
Theatre safe combination	The safe combination is: 8439.
The computer password	The computer password is UCUCC.
The computer action	Change the robot from special guard to hall patrol.
The museum dragon	Show the dragon the medallion.
The target sites	The Statue of Liberty, The Empire State Building, Bellevue Hotel Grand Central Station.
The space ship buttons	plug in the four modules, power on (upper left), middle square button, middle screen button, right screen button, right & left square buttons.

Advanced Playing Technique for ...

Wizardry V

Sir Tech

Use of a good sector editor (NU from Norton Utilities for example) can allow one to modify selected data to create a playable set of characters for Wizardry 5, rather than the 'wimps' one usually starts with. This gives a player a chance to play with 'decent' charactors. Data resides on sectors 92 - 93 -94 (a disk has sectors 0-719 decimal).

Contents (& Notes)	Char. 1	Char. 2	Char. 3	Char. 4	Char. 5	Char. 6
Sector #	92	92	92-93	93	93-94	94

(All of the following locations are in Hex)

	Char. 1	Char. 2	Char. 3	Char. 4	Char. 5	Char. 6
# Char in name	0	F6	1EC	1D8	CE	
Char name (15 MAX)	1-F	F7-105	1ED-1FB	E3-F1	1D9-1E7	CF-DD
# Char in password	10	106	1FC	F2	1E8	DE
Password name (15)	11-1F	107-115	1FD-B	F3-101	1E9-1F7	DF-ED
Race (1 TO 5)	20	116	C	102	1F8	EF
Class (0 TO 7)	22	118	E	104	1FA	F0
Alignment (1 TO 3)	24	11A	10	106	1FC	F2

lockbit (boolean), status, age (Integer), ability, saving throw (go here)

	Char. 1	Char. 2	Char. 3	Char. 4	Char. 5	Char. 6
# Gold (LO-HI BYTES)	34-35	12A-12B	20-21	116-117	C-D	102-103
Items held (8)	3A	130	26	11C	12	108
Actual items (8 bytes each)	3C-5B	132-151	28-47	11E-3D	14-33	10A-129
## Experience (Lo-Hi)	5C-5F	152-155	48-4B	13E-141	34-37	12A-12D
Previous level	62	158	4E	144	3A	130
Current level	64	15A	50	146	3C	132
### Current hit points	66-67	15C-15D	52-53	148-149	3E-3F	134-135
### Maximum hit Points	68-69	15E-15F	54-55	14A-14B	40-41	136-137

Spells known, m spell available, c spell available, fight level, armrclass, regen, poison, autokill, numswings, damage (go here)

Race: 1 = Human, 2 = Elf, 3 = Dwarf, 4 = Gnome, 5 = Hobbit
Class: 0 = Fighter, 1 = Mage, 2 = Priest, 3 = Thief, 4 = Bishop, 5 = Samuri, 6 = Lord, 7 = Ninja
Alignment: 1 = Good, 2 = Neutral, 3 = Evil

Good ones to change:	Hex	Decimal
# Gold	5555	21845
# Experience	55555	Lots
# Hit points	C7	199

Be careful, putting in too large a number makes the value negative.

Rocket Ranger codes

Country	Alge	Arab	Atla	Braz	Cana	Colu	Cong	E.Af	Egyp	Engl	Fran	Germ	Ital	Keny	Liby	Mide	Nige	Pers	Peru	Scan	Spai	Suda	USA	USSR	Vene	W.Af	Yugo
Algeria	0	28	20	43	37	44	25	29	21	16	14	17	15	31	10	19	23	30	50	22	13	24	36	26	40	18	12
Arabia	28	0	41	57	59	65	30	18	14	34	29	27	24	26	20	17	31	12	70	32	33	19	8	3	61	37	22
Atlantic	20	41	0	32	27	34	37	39	33	18	19	26	22	43	29	31	35	42	44	25	17	36	23	38	30	24	28
Brazil	43	57	32	0	49	27	38	48	52	50	45	53	47	46	42	54	3	61	25	55	40	44	41	60	28	33	51
Canada	37	59	27	49	0	28	56	62	51	31	36	3	39	64	44	47	48	55	40	34	35	57	10	41	26	42	43
Columbia	44	65	34	27	28	0	52	59	46	50	48	62	49	57	45	66	17	51	43	61	26	58	12	39	53		
Congo	25	30	37	38	56	52	0	22	23	36	33	34	32	17	24	26	15	31	54	40	29	20	55	39	50	21	27
E.Africa	9	18	39	48	62	64	22	0	15	37	32	33	30	14	23	24	25	19	67	35	31	12	61	28	60	34	26
Egypt	21	14	33	52	51	59	23	15	0	27	24	20	19	25	12	11	18	17	63	28	26	13	0	22	54	29	16
England	16	34	18	50	31	47	36	37	27	0	8	11	13	40	21	22	33	29	53	15	12	3	32	19	41	28	20
France	14	29	19	45	36	46	33	32	24	8	0	12	10	37	1	20	30	28	52	16	9	27	35	21	42	25	15
Germany	17	27	26	53	38	50	34	33	20	11	12	0	9	35	1	14	32	24	58	10	11	26	38	18	44	27	8
Italy	15	24	22	47	39	48	32	3	19	13	10	9	0	33	14	12	29	25	57	17	11	26	38	18	44	27	8
Kenya	31	26	43	46	64	62	17	14	25	40	37	35	33	0	28	30	22	27	65	39	36	18	63	34	59	32	29
Libya	0	20	29	42	4	49	24	23	12	21	17	18	14	28	0	13	19	22	56	27	15	16	43	25	46	26	11
Mideast	9	17	31	54	47	57	26	24	11	22	20	14	12	30	13	0	28	18	64	21	25	23	46	16	52	36	10
Nigeria	23	31	35	37	48	45	15	25	18	33	30	32	29	22	19	28	0	34	9	36	27	17	47	40	43	16	24
Persia	30	12	42	61	55	66	31	19	17	29	28	24	25	27	22	18	34	0	72	2	32	21	54	20	62	41	23
Peru	50	70	44	25	40	17	54	67	63	53	52	58	57	65	56	64	49	72	0	60	51	62	33	66	20	43	61
Scandanavia	22	32	25	55	34	51	40	3	28	15	16	10	17	39	27	21	36	26	60	0	20	31	37	14	47	38	18
Spain	13	33	17	35	43	29	31	26	12	9	16	11	36	15	25	27	32	51	20	0	28	34	24	39	23	19	
Sudan	24	19	36	44	57	61	20	12	13	30	27	29	26	18	16	23	17	21	62	31	28	0	56	32	55	22	25
USA	36	58	23	41	10	26	55	61	50	32	35	39	38	63	43	46	47	54	33	37	34	56	0	45	24	40	42
USSR	26	23	38	60	41	54	39	28	22	19	21	15	18	34	25	16	40	20	6	14	24	32	45	0	53	44	17
Venezuela	40	61	30	28	26	12	50	60	54	41	42	45	44	59	46	52	43	62	20	47	39	55	24	53	0	35	49
W.Africa	18	37	24	33	42	39	21	34	29	28	2	31	27	32	26	36	16	43	38	23	22	40	44	35	0	30	
Yugoslavia	12	22	28	51	43	53	27	26	16	20	15	13	8	29	11	10	24	23	61	18	19	25	42	17	49	30	0
Checksums:																											
Sum Vert	663	87	793	1158	1099	1203	866	888	7	723	682	709	667	955	670	736	803	891	1352	794	688	804	1068	838	1117	814	692
Sum Horiz	663	887	793	1158	1099	203	866	888	742	723	682	709	667	955	670	736	803	891	1352	794	688	804	1068	38	1117	814	692
	OK	O	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	O	OK	OK	OK	OK

Objective: Find the 5 secretly located rocket labs, obtain the components to build a rocket ship, capture enough lunarium to fly to the source of the Nazis' power, the Moon. Rescue Otto and Jane Barnstorff: (3 chances) 1 Zeppelin - destroy aerial torpedoes & board zeppelin. 2 Germany - reach bomb factory (destroy ME-19's first) & rescue them. 3 Desert Base - blast ack-ack batteries & rescue them.

---

unClassifieds

RDEX Contributors:

Most Wanted

Airheart	Broderbund
Alcon	Taito
Alien Mind	PBI Software
Apple Panic	Broderbund
Aquatron	Sierra
Arkanoid	Taito
Arkanoid II	Taito
Bad Street Brawler	Mindscape
Bank Street Writer Plus	Broderbund
Beyond Zork	Infocom
Bilestoad	Datamost
Border Zone	Infocom
Borg	Sirius
Bouncing Kamungas	Penguin
Boxing	?
Bubble Ghost IIgs	Accolade
Bureaucracy	Infocom
Centauri Alliance	Broderbund
Chuck Yeager's Advanced Flight Trainer	Electronic Arts
C'est La Vie	Adventure International
Comics	Accolade
Cosmic Relief	Datasoft
Crime & Punishment	Imagic
Crossword Magic v4.0	?
Darklord	Datasoft
Deathlord	Electronic Arts
Desecration	Mind Games
Designasaurus 128K	Britanica
Dig Dug	Thunder Mountain
Disk Optimizer System	Nibble Notch
Dondra	Spectrum Holobyte
D&D Master Assitant vol2	SSI
Dungeon Master (IIgs)	FTL
DROL	Broderbund
Eliminator	Adventure International
Epoch	Sirius
Explore-Australia	Dataflow Computer Service
Evolution	Sydney
Falcons	Piccadilly
Factastics Trivia	Daystar
Final Assault GS	Epyx
Force 7	Datasoft
Frogger	Main Street
GEOS	Berkley Softworks
Geometry GS	Broderbund
Gladiator	Taito
Goldrush	Sierra On Line
Gorgon	Sirius
GradeBuster 1 2 3	Grade Buster
Gutenbery Sr.	Micromation LTD.
Halls of Montezuma	Electronic Arts
High Orbit	Softsmith
Horizon V	Softsmith
Ice Demons	Morningstar
Impossible Mission GS	Epyx
Indoor Sports	Mindscape
Infocomics	Infocom
Jane	?
Joker Poker	Mindscape
Kingdom of Facts	Santa Barbara/Thunder Mountain
King of Chicago	Cinemaware
Lancaster	SVS
Legacy of the Ancients	?
Lost Tomb	Datasoft
Manhunter New York IIgs	Sierra On Line
Math Blaster Plus 3.5	Davidson
Mavis Beacon Teaches Typing (gs)	Software Toolworks
Microwave	Cavalier
Might and Magic II	Activision
Modem MGR	MGR Software
Mr. Pixel's Cartoon Kit	Mindscape/Thunder Mountain
Mr. Pixel's Programming Paint Kit	Mindscape/Thunder Mountain
National Inspirer	Tom Snyder Productions
Observatory (The)	Mindscape/Lightspeed Software
Odin	Odessta
Operation Wolf	Taito
Pensate	Datasoft/Softdisk
Phantoms 5	Sirius
Pig Pen	Datamost
Platoon	Data East
Project: Space Station	Avantage
Pulsar II	Sirius
Pure Stat Basketball	?
Quadratic Equations II	Olympus Educational Software
Questron II	Electronic Arts
Rails West	SSI
Rastan	Taito
Read n' Roll	Davidson
Rear Guard	Adventure International
Renegade	Taito
Rescue Raiders	Sir Tech
Rings of Saturn	Level 10
Risk	Leisure Games
Rocket Ranger (IIgs)	Cinemaware
S.D.I. (IIgs)	Cinemaware
Sailing Trough Story Problems	DLM/Neosoft
Sea Stalker	Broderbund
Serpentine	Broderbund
Skeletal System	Brainbank
Sky Shark	Taito
Sound Song & Vision	Advanced Software
Space Ark	Datamost
Spare Change	Broderbund
Spectre	Datamost
Speedy Spides	Readers Digest
Star Cruiser	Sirius
Star Maze	Sir Tech
StickyBear Math: Add & Subtract	Optimum Resources
Stickybear GS Versions 3.5	Xerox
Strike Fleet	Electronic Arts
Succession	Piccadilly
Superstar Ice Hockey	Mindscape
Superstar Indoor Sports	Mindscape
Talking Text Writer GS	Scholastic
Tangled Tales	Origin Systems
Test Drive	Accolade
Tetris (IIe)	Spectrum Holobyte
The Three Stooges (IIgs)	Cinemaware
Thunder Chopper	?
Ticket to Washington D.C.	Blue Lion Software
Tomahawk	Electronic Arts
Tomahawk (IIgs)	Datasoft
Triad	AI/Thunder Mountain
Trinity	Infocom
Ultima Trilogy	Origin Systems
Volcanoes v1.8	Earthware Comp. Services
War in the Middle Earth	Melbourne
Wasteland	Electronic Arts
Wayout	Sirius
Wings of Fury	Broderbund
Wizardry:Return of Werda	Sir-Tech.
Word Attack Plus (IIgs)	Davidson
Works (the)	First Star Software
Zenith	Softsmith
ZorkQuest	Infocom

IBM Most Wanted

Graphitti	George Best Phillips Academy
Gunship	Microprose
Heros of the Lance	SSI